

PYTHON BASIC PROGRAMS

Hello world in Python

INPUT FORMAT:

You do not need to read any input.

OUTPUT FORMAT:

Print Hello World on the output console.

SAMPLE OUTPUT:

```
Hello World
```

Algorithm for Hello World in Python

We clearly don't need an algorithm to print Hello World in Python. It can be printed using a simple print statement. (Print is a function in Python used to print output. Know more about printing the output in Python) Printing Hello World in Python is very easy when compared to other programming languages. **Here is a simple comparison of Hello World in different languages.**

Hello World Program in Python

Prerequisite Knowledge: In Python, a string should be enclosed in single or double or triple-quotes.

```
#Program to print hello world using single quotes print ('Hello World')  
  
#Program to print hello world using double quotes print ("Hello World")  
  
#Program to print hello world using triple quotes print ("Hello World")
```

Output:

```
Hello World  
Hello World  
Hello World
```

ADDING TWO NUMBERS IN PYTHON

Sol:

Algorithm to add two numbers in Python

Step 1: Get 2 inputs from the user. This can be done using the input() function (Here is how to take an input from the user). The input should strictly be of integer type. If in case, you want to take input of float type, then you need to typecast your input this way -float(input())

Step 2: Add these two inputs using arithmetic operators (+, -, /, /) or bitwise operators.

Step 3: Print the output

Methods to Add Two Numbers in Python

Method 1: Add two numbers using arithmetic operator (+)

```
#program to add two numbers in python using + arithmetic operator
a = int(input("Enter the first input: "))
b = int(input("Enter the second input: "))
print (a+b)
```

Input:

2
3

Output:

5

Method 2: Add two numbers using arithmetic operators * & -

```
#program to add two numbers in python using * and - arithmetic operators
a = int(input("Enter the first input: "))
b = int(input("Enter the second input: "))
c = (a*a-b*b)/(a-b)
print (c)
```

Input:

10
20

Output:

30

Method 3: Add two numbers in Python using decision making statements (if, else)

```
#program to add two numbers in python using if & else
a = int(input("Enter first input: "))
b = int(input("Enter second input: "))
if a!=b: #when a is not equal to b print (a+b)
else:
    print (2*a)
```

Input:

5
5

Output:

10

Method 4: Add two numbers in Python using functions

```
#add two numbers in python using functionsdef add(a, b): #user-defined function
    sum = a + b #adding 2 numbersreturn sum

#taking input from the user
a = int(input("Enter your first number: "))
b = int(input("Enter your second number: "))

#function call
print(add(a,b))
```

Input:

10
15

Output:

25

Method 5: Add two numbers using bitwise operators & functions

```
#program to add two numbers in python using bitwise operatorsdef
add_without_plus_operator(a, b):while b != 0:
    data = a & b
    a = a ^ b
    b = data << 1return a
```

```
a = int(input("Enter your first input: "))
b = int(input("Enter your second input: "))
print(add_without_plus_operator(a,b))
```

Input:

12

24

Output:

36

Method 6 (Additional): Add two float numbers using arithmetic operators

```
#program to add integer or float numbers
num1 = float(input("Enter your first input: "))
num2 = float(input("Enter your second input: "))

sum = num1 + num2
#You can print the sum in two ways
print(sum)

#or print ("The sum of {0} and {1} is {2}".format(num1, num2, sum))
```

Input:

15.5

26.4

Output:

41.9

The sum of 15.5 and 26.4 is 41.9

Method 7: Add two numbers in one line

```
#add two numbers using python in one line
print("The sum is %.1f"%(float(input('Enter first number: ')) + float(input('Enter second number: '))))
```

Input:

67

21

Output:

41.9

The sum is 88.0

AREA OF A CIRCLE IN PYTHON

Algorithm to Calculate the Area of Circle in Python

Step 1: Get an input from the user using the input () function. Here the input corresponds to the radius of the circle.

Step 2: Calculate the area of a circle by using the formula $\text{area} = \pi r^2$

Step 3: Print the output i.e area of the circle.

Methods To Find Area of Circle in Python

Method 1: Area of circle by importing math file

```
#program to find area of circle in Python using math file
import math
r = float(input("Enter the radius of the circle: "))
area = math.pi* r * r
print("%.2f" %area)
```

Input:

6

Output:

113.10

Method 2: Area of Circle using?

```
#program to find area of circle in Python using π
PI = 3.14
r = float(input("Enter the radius of the circle: "))
area = PI * r * r
print("%.2f" %area)
```

Input:

8

Output:

200.96

Method 3: Area of Circle using functions

```
#program to find area of circle using functions
import math
```

```
#Function that calculates area of circle
def area_of_circle(r):
    a = r**2 * math.pi
    return a

r = float(input("Enter the radius of the circle: "))
print("%.2f" %area_of_circle(r))
```

Input:

3

Output:

28.27

Conclusion

Among the 3 methods discussed above to calculate the area of circle in Python, **the first one is the easiest**. This is because by importing the math file we can directly get the pi value and this makes the calculation simpler.

SWAP TWO VARIABLES WITH & WITHOUT TEMPORARY VARIABLE IN PYTHON

Algorithm To Swap Two Numbers in Python

Step 1: Get two integer inputs x and y from the user using the input() function.

Step 2: Now, using the third variable, swap x & y using arithmetic or bitwise operations in order to avoid third/temporary variable.

Step 3: Print the output i.e the swapped values

Swapping Two Variables in Python Using Various Methods

Method 1: Swap two numbers using third variable

```
#Python program to swap two variables#Taking input from the user
x = int(input("Enter the first number: "))
y = int(input("Enter the second number: "))

#create a temporary variable and swap the values
temp = x
x = y
y = temp
```

```
print("Value of x is %d" %x)
print("Value of y is %d" %y)
```

Input:

23

Output:

32

Method 2: Swap two numbers using arithmetic operators + & -

Prerequisite: Arithmetic operators in Python

```
#Python program to swap two variables without using third variable#taking the input from the user
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))

#swapping using arithmetic operators + & -
a = a + b
b = a - b
a = a - b

#printing the output
print("Swapped values of x is %.1f & y is %.1f" %(a,b))
```

Input:

5.8

2.6

Output:

2.6

5.8

Method 3: Swap two numbers using arithmetic operators * & /

```
#Python program to swap two variables without using third variable

a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))

#swapping using arithmetic operators * & /
a = a * b
```

```
b = a / b
a = a / b

#printing the output
print("Swapped value of x is %d & y is %d" %(a,b))
```

Input:

54
99

Output:

99
54

Method 4: Swap two numbers using bitwise operators

Prerequisite: Bitwise operators in Python

```
#Python program to swap two variables without using third variable

a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))

#swapping using bitwise operators
a = a ^ b
b = a ^ b
a = a ^ b

#printing the output
print("Swapped value of x is %d & y is %d" %(a,b))
```

Input:

548
847

Output:

847
948

Method 5: Swap two numbers using functions

Prerequisite: Functions in Python


```
#Python program to swap two variables without using third variable
def swap(a, b):
    t = a
    a = b
    b = t
    return a, b

#getting input from the user
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))

#calling the function
a, b = swap(a, b)

#printing the output
print("Swapped value of x is %d & y is %d" %(a,b))
```

Input:

54.2648
25.1239

Output:

25.12
54.26

Conclusion

Though there are different methods to swap two variables, **method 2 is the easiest of all**. It occupies **less memory space** compared to the other methods.

FACTORIAL OF A NUMBER IN PYTHON

Factorial of a number n (non-negative) is the product of all positive numbers less than or equal to n. In this article, we will focus on factorial program in Python i.e calculating the factorial of a given number. So, before we have a look at the algorithm, quickly have a look at the input-output.

INPUT FORMAT:

A single line of input containing a positive integer for which the factorial has to be calculated.

OUTPUT FORMAT:

A single line of output containing the factorial of the inputted number

SAMPLE INPUT:

5

SAMPLE OUTPUT:

120

PREREQUISITE KNOWLEDGE: Loops in Python

ALGORITHM FOR FACTORIAL PROGRAM IN PYTHON

Below is the step by step algorithm for factorial program in Python.

Step 1:Get a positive integer input (n) from the user.

Step 2:Iterate from 1 to n using a for loop (for loop is used to increment the number up to the given input)

Step 3:Using the below formula, calculate the factorial of a number $f = f * i$

Step 4:Print the output i.e the calculated factorial

FACTORIAL OF A NUMBER IN PYTHON**Method 1: Factorial Program in Python using For Loop**

Prerequisite:For loop in Python

```
#Python program to print factorial of a number
num = int(input("Enter the number: "))
f = 1#iterating through the num valuefor i in range (1, num+1):
    f = f*i

#printing the output
print ("Factorial of the number %d is %d" %(num, f))
```

Input:

4

Output:

24

Method 2: Factorial Program in Python using If Else

Prerequisite:If else Decision Making Statements in Python

```
#Python program to find factorial of a number
num = int(input("Enter a number: "))
factorial = 1#check if the number is negative, positive or zero
if num < 0:
    print("Invalid Input")
elif num == 0:
    print ("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print ("Factorial of %d is %d" %(num, factorial))
```

Input:

6

Output:

720

Method 3: Factorial Program in Python using Recursion

Prerequisite:Recursion in Python

```
#Python program to print factorial of a number#defining a recursive function
def factorial(num):
    if num == 1:
        return num
    else:
        return num * factorial(num - 1)

#getting input from the user
num = int(input("Enter the number: "))
if num < 0: #if input number is negative then return an error message
    print("Invalid input")
elif num == 0: #elif the input number is 0 then display 1 as output
    print ("Factorial of 0 is 1")
else: #else calculate the factorial by calling the user defined function
    print ("Factorial of %d is %d" %(num, factorial(num)))
```

Input:

3

Output:

6

RANDOM NUMBERS IN PYTHON

we will see how to generate random numbers in Python. Python has a set of functions that are used to generate or manipulate random numbers.

These functions are used in lottery games and other applications requiring random number generation. So, before we have a look at these functions, quickly have a look at the input-output.

INPUT FORMAT:

A single line of input containing a positive integer that indicates how many random numbers must be generated.

OUTPUT FORMAT:

A list consisting of random numbers.

SAMPLE INPUT:

3

SAMPLE OUTPUT:

[8, 9, 1]

PREREQUISITE KNOWLEDGE: Modules in Python and Built-In Modules in Python

Algorithm to Generate Random Numbers from 1 to 20 and Append them to a List

Step 1: Firstly, Import the random module into the program.

Step 2: Then, get the number of elements from the user as an integer input using `int(input())` function.

Step 3: Now, using a for loop and **`random.randint()`** function, generate random numbers. There are two arguments in this function which denote the range (start, end).

Step 4: And then, append it to a list using the `append()` function.

Step 5: Print the appended list.

Methods to Generate Random Numbers in Python

Method 1: Random Numbers in Python using `randint()`

```
#program to generate random numbers in python
import random
a=[]
```

```
n=int(input("Enter the number of elements:"))
for j in range(n):
    a.append(random.randint(1,20)) #the random numbers are appended to the list 'a' using
append().#printing the output
print('Randomised list is: ',a)
```

Input:

Enter the number of elements: 7

Output:

Randomised list is: [16, 18, 19, 18, 6, 4, 12]

Method 2: Random Numbers using choice () and randrange ()

In the below program, we have used two functions to generate Random Numbers in Python. Let's see how that works. The function **choice()** is used to generate 1 random number from a container.

The function **randrange(beg, end, step)** is also used to generate random numbers but within a range specified in its arguments. This function takes 3 arguments,

- **beg** - beginning number (included in generation),
- **end** - last number (excluded in generation) and
- **step** - used to skip numbers in range while selecting.

```
#Python Program to Generate Random Numbers using choice() and randrange()import random
print ("Random number from list is ",end="")

#using choice() to generate a random number from a given list of numbers.print
(random.choice([1, 4, 8, 10, 3]))

#using randrange() to generate a random number within 20 and 50 with step size of 3.print
("Random number from range is ",end="")
print (random.randrange(20, 50, 3))
```

Output:Random number from list is 3

Random number from range is 23

Method 3: Random Numbers using random () and seed ()

Here, the functions such as random() and seed() are used to generate Random Numbers in Python.

This function **random()** is used to generate a floating-point random number that is less than 1 and is greater or equal to 0.

The function **seed()** maps a particular random number with the seed argument mentioned. The Random numbers that are called after the seeded value, Returns the mapped number.

```
#Python Program to Generate Random Numbers using random() and seed()import random
```

```
# using random() to generate a random number between 0 and 1 print ("Random number
between 0 and 1 is ", end="")
print (random.random())

# using seed() to seed a random number
random.seed(5)

# printing mapped random number print ("The mapped Random Number with 5 is ", end="")
print (random.random()))
```

Output:Random number between 0 and 1 is 0.1038744038279239
The mapped Random Number with 5 is 0.6229016948897019

Method 4: Random Numbers using shuffle () and uniform ()

In this method, functions like shuffle() and uniform() are used to generate Random Numbers in Python.

This function **shuffle()** is used to shuffle the entire list to randomly arrange them.

This function **uniform(a, b)** is used to generate a floating-point random number between the numbers mentioned in its arguments. It takes two arguments,

- **a** - lower limit (included in generation)
- **b** - upper limit (not included in generation).

```
# # Python Program to generate Random Numbers using shuffle() and uniform(a, b)import
random
# Initializing list
li = [1, 4, 5, 10, 2]
# Printing list before shuffling print ("Before shuffling, The list is", end="")
for i in range(0, len(li)):
    print (li[i], end=" ")
print("\r")
random.shuffle(li) # using shuffle() to shuffle the list print ("After shuffling, The list becomes
", end="") # Printing list after shuffling for i in range(0, len(li)):
    print (li[i], end=" ")
print("\r")
print ("The random floating point number between 5 and 10 is : ",end="")
# using uniform() to generate random floating number between 5 and 10 print
(random.uniform(5,10))
```

Output:The list before shuffling is : 1 4 5 10 2
The list after shuffling is : 4 5 2 10 1
The random floating-point number between 5 and 10 is : 9.568792561810309
These are different methods to generate Random Numbers in Python.

LCM OF TWO NUMBERS IN PYTHON

LCM can be referred to as the **Least Common Multiple** of two numbers. For example, the LCM of two numbers a and b is the smallest positive integer that is divisible by both.

Before we look at the algorithm, Let's have a look at the input and output format.

INPUT FORMAT:

Input consists of 2 integers

OUTPUT FORMAT:

Print the LCM of a given number

SAMPLE INPUT:

```
12
24
```

SAMPLE OUTPUT:

```
84
```

Algorithm for LCM of 2 numbers in Python

Step 1:Initially, Get 2 Integer Inputs from the user using **int(input())**.

Step 2:Find the greater number by using an If condition and assign it to the variable 'max'.

Step 3:Within the while loop, Use an If condition to check whether the remainder of (max% a) and (max% b) equals to zero or not.

Step 4:If true, Print max which is the LCM of 2 numbers,

Step 5:Otherwise, skip that value using a **break** statement.

Step 6:End of the program

Methods to find the LCM of Two Numbers in Python

Method 1: LCM of 2 numbers

```
# Python Program to find the LCM of Two Numbers
num1 = int(input())
num2 = int(input())
if(num1 > num2 ): # Use If condition to find the greatest number among these two.
    max= num1
else:
    max= num2
while(True):
    if(max % num1 == 0 and max % num2 == 0):
        print(max)
```

```
break;
max= max+ 1
```

Input:

12
24

Output:

24

Method 2: LCM of Two Numbers using Functions

```
# Python Program to find LCM of Two Numbers using Functions
def findlcm(a, b): # Function definition
    if(a > b):
        maximum = a
    else:
        maximum = b
    while(True):
        if(maximum % a == 0 and maximum % b == 0):
            lcm = maximum;
            break;
        maximum = maximum + 1
    return lcm
# Taking inputs from the user
num1 = int(input())
num2 = int(input())
lcm = findlcm(num1, num2) # Function call
print(lcm)
```

Input:

12
26

Output:

256

Method 3: LCM of Two Numbers using GCD

```
# Python Program to find the LCM of Two Numbers using GCD
num1 = int(input())
num2 = int(input())
a = num1
b = num2
while(num2 != 0): # When num2 is not equal to 0 # Swap num2 with num1 using temp
    temp = num2
    num2 = num1 % num2 # Modulus of num1 and num2
```



```
num1 = temp
gcd = num1
lcm = (a * b) // gcd # LCM is found using GCD
print(lcm)
```

Input:

3

8

Output:

24

Method 4: LCM of Two Numbers using Recursion

```
# Python Program to find the LCM of Two Numbers using Recursion
def findgcd(a, b): # Function definition
    if(b == 0):
        return a;
    else:
        return findgcd(b, a % b) # Recursion takes place here
num1 = int(input())
num2 = int(input())
gcd = findgcd(num1, num2) #function call
lcm = (num1 * num2) // gcd
print(lcm)
```

Input:

28

56

Output:

56

So, These are the various methods that can be used to find the LCM of two numbers in Python. The simplest of all is method 1.

GCD OF TWO NUMBERS IN PYTHON

How to find the GCD of two numbers in Python programming. GCD (Greatest Common Divisor) or HCF (Highest Common Factor) is the largest positive integer that divides each of the integers without leaving any remainder.

GCD of Two Numbers in Python

Factors of 24 are 2, 2, 2, 3

Factors of 18 are 2, 3, 3

On Multiplying common factors = $2 * 3$
we get,

GCD of 24 and 18 = 6

Before we look at the algorithm, have a look at the sample input/output.

INPUT FORMAT:

Input consists of two integers for which GCD is to be calculated

OUTPUT FORMAT:

Output returns one argument, which is the GCD.

SAMPLE INPUT:

25

5

SAMPLE OUTPUT:

5

Algorithm to Find the GCD of Two Numbers in Python

Step 1:Get 2 integer inputs from the user

Step 2:Next, use decision-making statements or use recursion to check if both the given numbers are divisible by any number (i) without leaving any remainder.

Step 3:If true, then $GCD = i$

Step 4:Print the GCD of the two numbers

Step 5:End of the program

GCD of Two Numbers

Method 1: GCD of Two Numbers Using While loop

```
# Python Program to find GCD of Two Numbers using While loop
num1 = int(input("Enter 1st number: "))
num2 = int(input("Enter 2nd number: "))
i = 1 while(i <= num1 and i <= num2):
    if(num1 % i == 0 and num2 % i == 0):
```

```
gcd = i
i = i + 1
print("GCD is", gcd)
```

Input:

5
15

Output:

5

Method 2: GCD of 2 numbers using Recursion

```
#Python Program to find GCD of Two Numbers using Recursion
def gcd(a,b):if(b==0):
    return a
else:
    return gcd(b,a%b)
a=int(input())
b=int(input())
GCD=gcd(a,b)
print(GCD)
```

Input:

12
36

Output:

12

Method 3: GCD of 2 numbers using a temp variable

```
# Python Program to find GCD of Two Numbers a temp variable
num1 = int(input())
num2 = int(input())
a = num1
b = num2
while(num2 != 0):
    # swap using temp variable
    temp = num2
    num2 = num1 % num2
    num1 = temp
gcd = num1
print(gcd)
```

Input:

15

75

Output:

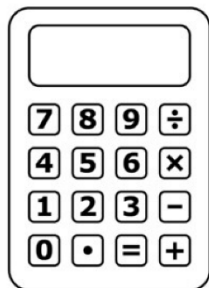
15

So, these are the various methods used to find the GCD of Two Numbers in Python.

CALCULATOR PROGRAM IN PYTHON

How to build a Simple Calculator in Python which will perform various **Calculator operations** such as **Addition, Subtraction, Multiplication, Division**.

Calculator Program in Python



$$2 + 3 = 5$$

$$5 - 2 = 3$$

$$3 * 2 = 6$$

Before we get into the algorithm, Let's have a look at the Input, Output format

.

INPUT FORMAT:

- The first line of input corresponds to the choice of operation
- The second line of input consists of 2 integers

OUTPUT FORMAT:

Perform the operation and print the output

SAMPLE INPUT:

Enter your choice: 2

Enter two numbers:

9

5

SAMPLE OUTPUT:

Result = 4

PREREQUISITE KNOWLEDGE:If-Else in Python

Algorithm for calculator program in Python

Step 1:Get input from the user (choice corresponding to various operations like addition, subtraction, multiplication, division)

Step 2:Get two integer inputs from the user.

Step 3:Perform the operations based on the user's choice.

Step 4:Print the output

Step 5:End the program

Prerequisite Knowledge:We clearly know that a calculator performs different types of arithmetic operations. For performing a particular task, We should get the choice from the user. For this, we should have an idea about Nested if statements in Python.

Methods to build a simple calculator in Python

Method 1: Simple Calculator using Nested If

```
# Python Program to build a Simple Calculator using Nested If
choice = int(input("Enter your choice:n")) # User's choice[1,2,3,4]if (choice>=1 and
choice<=4):
    print("Enter two numbers: ")
    num1 = int(input())
    num2 = int(input())

    if choice == 1: # To add two numbers
        res = num1 + num2
        print("Result = ", res)

    elif choice == 2: # To subtract two numbers
        res = num1 - num2
        print("Result = ", res)

    elif choice == 3: # To multiply two numbers
        res = num1 * num2
        print("Result = ", res)

    elif choice == 4: # To get quotient with decimal value
        res = num1 / num2
        print("Result = ", res)

    elif choice == 5:
        exit()
    else:
```

```
print("Wrong input...!!")
```

Input:

Enter your choice:

3

Enter two numbers:

9

5

Output:

Result = 45

Method 2: Calculator program using function

```
# Function Definition
def calculate():
    print(+)
    print(-)
    print(*)
    print(/)
    operation = input("Select an operator:n")
    print("Enter two numbers")
    number_1 = int(input())
    number_2 = int(input())

    if operation == '+': # To add two numbers
        print(number_1 + number_2)

    elif operation == '-': # To subtract two numbers
        print(number_1 - number_2)

    elif operation == '*': # To multiply two numbers
        print(number_1 * number_2)

    elif operation == '/': # To divide two numbers
        print(number_1 / number_2)

    else:
        print('Invalid Input')
# Function Call
calculate()
```

Input:

Enter your choice:

5

Enter two number:

25

5

Output:


Invalid Input

The above methods are the ones that can be used to build a Simple Calculator in Python.

SUM OF NATURAL NUMBERS IN PYTHON

How to find the sum of natural numbers in Python programming. A Natural number can be defined as a whole, non-negative number in the number system.

Sum of natural numbers in Python



A diagram illustrating the range of natural numbers. It shows the number 1 on the left and the infinity symbol (∞) on the right, connected by a horizontal red line. Below this line is a black curly bracket, indicating the span from 1 to infinity.

Natural Numbers = Whole, +ve numbers

We can find the sum of natural numbers by using loops or by using Recursion. So, before we look at the algorithm, Let's have a look at the input, output format.

INPUT FORMAT:

Input consists of 1 integer which denotes the number of Natural Numbers to be added

OUTPUT FORMAT:

Sum the values and print the output

SAMPLE INPUT:

5

SAMPLE OUTPUT:

15

Algorithm to print sum of natural numbers in Python

Step 1:Get an input from the user

Step 2:Check whether the given input is greater than zero or not

Step 3: If the input is greater than zero, then add the numbers using a while loop

Step 4: Print the output

Step 5: End the program

Methods to find the sum of Natural Numbers in Python

Prerequisite Knowledge: If-Else in Python

Method 1: Sum of natural numbers

```
# Python program to find the sum of natural numbers
num = int(input())
if num < 0:
    print("Enter a positive number")
else:
    sum = 0 # while loop to iterate from num till 0
    while(num > 0):
        sum += num
        num -= 1 # decrement num on each iteration
    print(sum)
```

Input:

5

Output:

15

Method 2: Sum of Natural Numbers using Recursion

```
# Python program to find the Sum of Natural Numbers up to n using Recursion# Function
declaration
def recur_sum(n):
    if n <= 1:
        return n
    else:
        return n + recur_sum(n-1) # Recursive Function
num = int(input())

if num < 0:
    print("Enter a positive number")
else:
    print(recur_sum(num)) # Function Call
```

Input:

10

Output:

55

Above are the two methods with which we can find the sum of Natural Numbers.

LEAP YEAR IN PYTHON

A leap year is defined as a calendar year containing one additional day (Feb 29th) added to keep the calendar year synchronized with the astronomical year.

Leap Year program in Python

No. of Days



Let's have a look at the input, output format before jumping into the algorithm.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT

String output showing leap year or non-leap year

SAMPLE INPUT

2004

SAMPLE OUTPUT

Leap year

Algorithm to check whether the given year is a Leap Year or not in Python

Step 1:Get input from the user

Step 2:Import the calendar module

Step 3:By using isleap() function or nested if or if condition, Check whether the given year is a leap year or not

Step 4:Print the output as Leap Year or Not a Leap year

Step 5:End the program

Methods to check for Leap Year in Python

Method 1: Leap year program by importing math module in Python

```
# Python program to check for Leap Year by importing math module in Python# importing
calendar module for calendar operations
import calendar
year = int(input())
# calling isleap() method to check for Leap Year
val = calendar.isleap(year)
if val == True:
    print("%s is a Leap Year" % year)
else:
    print("%s is not a Leap Year" % year)
```

Input:

2005

Output:

2005 is not a Leap Year

Method 2: Leap year program by using nested if condition in Python

```
# Python program to check for Leap Year using nested if
year = int(input())
if (year % 4) == 0:
    if (year % 100) == 0:
        if (year % 400) == 0:
            print("{0} is a leap year".format(year))
        else:
            print("{0} is not a leap year".format(year))
    else:
        print("{0} is a leap year".format(year))
else:
    print("{0} is not a leap year".format(year))
```

Input:

2000

Output:

2000 is a Leap Year

Method 3: Leap year program by using if condition in Python

```
# Python Program to Check Leap Year using If condition
year = int(input( ))
if (( year%400 == 0)or (( year%4 == 0 ) and ( year%100 != 0))):
    print("%d is a Leap Year" %year)
else:
    print("%d is Not the Leap Year" %year)
```

Input:

2012

Output:

2012 is a Leap Year

So, We have discussed three different methods to check whether the given year is a Leap Year or not in Python.

REVERSE A NUMBER IN PYTHON

Reverse a Number in Python



Before getting to the algorithm, let's have a look at the input, output format.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

Reverse the number and print the output

SAMPLE INPUT:

321

SAMPLE OUTPUT:

123

Algorithm to Reverse a number using while loop in Python

Step 1:Get input from the user

Step 2:Assume 0 to the variable "rev"

Step 3:Check whether the given number is greater than zero using while loop.

Step 4:If yes, find the remainder by performing modulus of 10 with the input.
Step 5:Multiply rev by 10 and add the remainder to it, store the answer in rev.
Step 6:Get the quotient of the input.
Step 7:The loop will repeat until the number is reversed.
Step 8:Print the output
Step 9:End the program.

Methods to Reverse a number in Python

Method 1: Reverse a number using while loop in Python

```
# Python program to Reverse a number in Python
num = int(input())
rev = 0
while num > 0:
    rem = num % 10 # Finding the remainder
    rev = (rev*10) + rem
    num = num//10 # Finding the quotient
print("%d " %rev)
```

Input:

1542

Output:

2451

Method 2: Reverse a number using while loop with function in Python

```
# Python Program to reverse a number using Functions
def Reverse(Number): # Function Definition
    rev = 0
    while(Number > 0):
        rem = Number %10
        rev = (rev *10) + rem
        Number = Number //10
    return rev

Number = int(input())
rev = Reverse(Number) # Function Call
print("%d" %rev)
```

Input :

2148

Output:

8421

Method 3: Reverse a number using while loop with recursion in Python

```
# Python program to reverse a number using recursion
rev_num = 0
base = 1
def reverse_digits(num): # Function Definition
    global base
    if(num > 0):
        reverse_digits((int)(num / 10))
        rev_num += (num % 10) * base
        base *= 10
    return rev_num

num = int(input())
print("Reversed number is", reverse_digits(num)) # Function call
```

Input:

1907

Output:

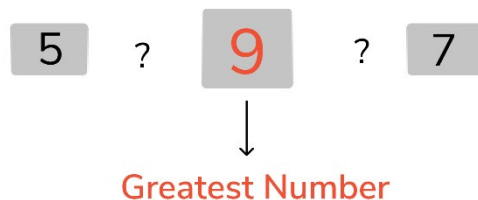
7091

The various methods to reverse a number in Python are discussed above. Method 1 is the easiest of all.

GREATEST OF THREE NUMBERS IN PYTHON

How to find the greatest of three numbers in Python.

Greatest of 3 Numbers in Python



Before getting to the algorithm, let's have a look at the input, output format.

INPUT FORMAT:

Input consists of 3 integers

OUTPUT FORMAT

Find the greatest number and print the output

SAMPLE INPUT:

44

12

76

SAMPLE OUTPUT:

76

PREREQUISITE KNOWLEDGE: If-Else in Python

Algorithm to find the greatest of three numbers using if-else statement in Python

Step 1:Get 3 inputs from the user

Step 2:Use an if-else statement, If (num1 > num2) and (num1 > num3), Print num1

Step 3:Else if (num2 > num1) and (num 2 > num3), Print num2

Step 4:Else, print num3

Step 5:End the program

Methods to find the greatest of three numbers using Python

Method 1: The greatest of three numbers using if-else statement in Python

```
# Python program to find the greatest of three numbers using if-else statement
num1 = int(input())
num2 = int(input())
num3 = int(input())
if (num1 >= num2) and (num1 >= num3):
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    greatest = num2
else:
    greatest = num3
print("The greatest number is",greatest )
```

Input:

54

86

24

Output:

86

Method 2: The greatest of three numbers using max function in Python

```
# Python program to find the greatest of three numbers using max function
num1 = int(input())
num2 = int(input())
```

```
num3 = int(input())  
print(max(num1,num2,num3),"is the greatest") # Built-in function "max"
```

Input:

24
58
95

Output:

95 is greater

Method 3: The greatest of three numbers using functions

```
# Python program to find the greatest of three numbers using functions  
def maximum(a, b, c): # Function Definition  
    if (a >= b) and (a >= c):  
        largest = a  
    elif (b >= c):  
        largest = b  
    else:  
        largest = c  
    return largest  
  
a = int(input())  
b = int(input())  
c = int(input())  
print(maximum(a, b, c)) # Function Call
```

Input:

256
24
123

Output:

256

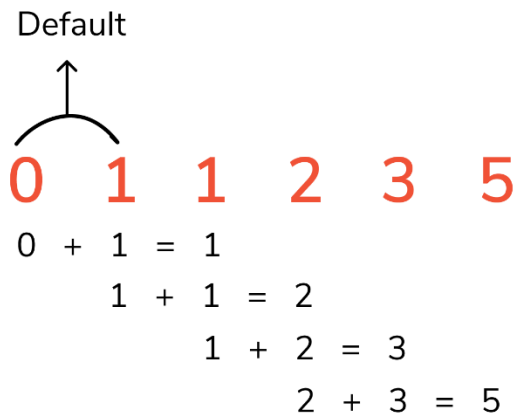
The greatest of three numbers in Python can be found by using any of the above methods. Method 2 occupies minimal space.

FIBONACCI SERIES IN PYTHON

How to write Python Programs to generate the Fibonacci series until the given input value using two different methods. Fibonacci series is a series in which each number is the sum of the preceding two numbers. By default, the first two numbers of a Fibonacci series are 0 and 1.

Fibonacci Series In Python | Python Program To Print Fibonacci Series

Fibonacci Series



INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

A single line of output containing the Fibonacci series until 7 values.

SAMPLE INPUT:

7

SAMPLE OUTPUT:

0 1 1 2 3 5 8

PREREQUISITE KNOWLEDGE: while loop in Python and Recursion in Python

Method 1: Using a while loop

Algorithm for printing Fibonacci series using a while loop

Step 1: Input the 'n' value until which the Fibonacci series has to be generated

Step 2: Initialize sum = 0, a = 0, b = 1 and count = 1

Step 3: while (count <= n)

Step 4: print sum

Step 5: Increment the count variable

Step 6: swap a and b

Step 7: sum = a + b

Step 8:while (count > n)

Step 9:End the algorithm

Step 10:Else

Step 11:Repeat from steps 4 to 7

Python program to print Fibonacci series until 'n' value using while loop

```
#Python program to generate Fibonacci series until 'n' value
n = int(input("Enter the value of 'n': "))
a = 0
b = 1
sum = 0
count = 1
print("Fibonacci Series: ", end = " ")
while(count <= n):
    print(sum, end = " ")
    count += 1
    a = b
    b = sum
    sum = a + b
```

Input:

Enter the value of 'n': 5

Output:

Fibonacci Series: 0 1 1 2 3

Method 2: Using recursion

Recursive function algorithm for printing Fibonacci series

Step 1:If 'n' value is 0, return 0

Step 2:Else, if 'n' value is 1, return 1

Step 3:Else, recursively call the recursive function for the value (n - 2) + (n - 1)

Python program to print Fibonacci series until 'n' value using recursion

```
#Python program to generate Fibonacci series Program using Recursion
def Fibonacci_series(Number):
    if(Number == 0):
        return 0
    elif(Number == 1):
        return 1
    else:
        return (Fibonacci_series(Number - 2) + Fibonacci_series(Number - 1))
```

```
n = int(input("Enter the value of 'n': "))
print("Fibonacci Series:", end = ' ')
for n in range(0, n):
    print(Fibonacci_series(n), end = ' ')
```

Input:

Enter the value of 'n': 5

Output:

Fibonacci Series: 0 1 1 2 3

PRIME NUMBER IN PYTHON

How to check if the given number is prime or not using two different methods. A prime number is a natural integer number which is greater than 1 and has no positive divisors other than the value 1 and itself.

In other words, a prime number is divisible by the value 1 and itself.

Prime Number

1 2 3 4 5 6 7 9 10 11 12 13.....

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

A single line saying "Prime Number", if the given input number is a prime value. Else, "Not a Prime Number"

SAMPLE INPUT:

11

SAMPLE OUTPUT:

Prime Number

PREREQUISITE KNOWLEDGE: for loop in Python and Functions in Python

Method 1: Using a 'for' loop

Algorithm to check if the given number is prime or not using for loop

Step 1:Get input from the user

Step 2:Check if the input number is greater than 1 or not

Step 3:If yes, iterate from the value 2 to the given input value using 'for' loop

Step 4: Check if the value is divisible by 1 and itself.

Step 5: If yes, print "Prime Number"

Step 6: Else, print "Not a Prime Number"

Step 7: Else, print "Enter a positive number"

Step 8: End

Python program to check if the given number is prime or not using for loop

```
#Python program to check if the given number is prime or not
num = int(input("Enter the number: "))
if(num > 1):
    #Iterate from 2 to num
    for i in range(2, num):
        if((num % i) == 0):
            print("Not a Prime Number")
            break
    else:
        print("Prime Number")
else:
    print("Enter a positive number")
```

Input:

Enter the number: 10

Output:

Not a Prime Number

Method 2: Using function

Python program to check if the given number is prime number or not using function

```
#Python Program to check if the given number is prime or not using function
def finding_prime(Number):
    count = 0
    for i in range(2, Number):
        if(Number % i == 0):
            count = count + 1
    return count

n = int(input("Enter the Number: "))
res = finding_prime(n)
if(res == 0 and n != 1):
    print("Prime Number")
else:
    print("Not a Prime Number")
```

Input:

Enter the number: 5

Output:

Prime Number

PRIME NUMBERS WITHIN THE RANGE IN PYTHON

How to print all the prime numbers that lie within the given input range.

INPUT FORMAT:

Input consists of two integers. One is the lower range and another is the upper range.

OUTPUT FORMAT:

A single line consisting of all prime numbers that lie within the given lower and upper range.

SAMPLE INPUT:

```
2
10
```

SAMPLE OUTPUT:

```
2 3 5 7
```

Algorithm to print prime numbers within the given range in Python

Step 1: Get 2 inputs (lower and upper range) from the user

Step 2: Using for loop iterate through the given range

Step 3: Check whether the values within the given range are divisible by 1 and itself

Step 4: If yes, print the value

Step 5: End

Python program to print prime numbers within the given range

```
low = int(input("Enter lower range: "))
high = int(input("Enter upper range: "))
for num in range(low, high + 1):
    if(num > 1):
        for i in range(2,num):
            if (num % i) == 0: #checking whether the value is not primebreakelse:
                print(num, end = " ")
```

Input:

Enter lower range: 3

Enter upper range: 15

Output:

```
3 5 7 11 13
```

PALINDROME PROGRAM IN PYTHON

How to check if the given number or a word is a palindrome or not. The palindrome is a word, phrase or number that reads the same backward as forward, i.e., any number or a word is said to be a palindrome if the original value and the reverse of it are the same.

First, let us check whether the given number is a palindrome or not using two different methods.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

A single line saying "Palindrome", if the given input number is palindrome. Else, "Not a Palindrome".

SAMPLE INPUT:

453

SAMPLE OUTPUT:

Not a Palindrome

Algorithm to check whether the given number is Palindrome or not in Python

Step 1: Input the number

Step 2: Find the reverse of the number

Step 3: If the reverse of the number is equal to the number, then print "Palindrome"

Step 4: Else, print "Not a Palindrome"

Method 1: Using a while loop

Python program to check if the given input number is a palindrome or not using a while loop

```
n = int(input("Enter the number : "))
reverse = 0
number = n
#reversing the given numberwhile(n != 0):
    remainder = n % 10
    reverse = reverse * 10 + remainder
    n = int(n / 10)
```

```
if(number == reverse):  
    print("Palindrome")  
else:  
    print("Not a Palindrome")
```

Input:

Enter the number: 454

Output:

Palindrome

Method 2: Using a function

Python program to check if the given input number is a palindrome or not using a function

```
#program to check whether the given input number is palindrome or not using a function  
def rev(temp):  
    remainder = 0  
    reverse = 0  
    while(temp != 0):  
        remainder = temp % 10  
        reverse = reverse * 10 + remainder  
        temp = int(temp / 10)  
    return reverse  
  
#main() function  
n = int(input("Enter the number: "))  
temp = n  
res = rev(temp)  
if (res == n):  
    print("Palindrome")  
else:  
    print("Not a Palindrome")
```

Input:

Enter the number: 759

Output:

Not a Palindrome

Next, let us check whether the given string is a palindrome or not using two different methods.

INPUT FORMAT:

Input consists of a string

OUTPUT FORMAT:

A single line saying "Palindrome", if the given input string is palindrome. Else, "Not a Palindrome".

SAMPLE INPUT:

MADAM

SAMPLE OUTPUT:

Palindrome

Algorithm to check whether the given string is Palindrome or not in Python

Step 1: Get the string from the user

Step 2: Find the reverse of the string

Step 3: If the reverse of the string is equal to the given string, then print Palindrome. Else, print Not a Palindrome

Method 1: Using a slice operator

Python program to check if the given input string is a palindrome or not using a slice operator

```
myStr = input("Enter the string: ")
if str(myStr) == str(myStr)[::-1] #checking using slicing operator
    print("Palindrome")
else:
    print("Not a Palindrome")
```

Input:

Enter the string: Welcome

Output:

Not a Palindrome

Method 2: Using a while loop

Python program to check if the given input string is a palindrome or not using a while loop

```
#program to check if the given input string is a palindrome or not using a while loop
myStr = input("Enter the string: ")
j = myStr
rev = ""while(len(j) > 0):
```

```
if(len(j) > 0):
    a = j[-1]
    j = j[:-1]
    rev = rev + a
if(rev == myStr):
    print("Palindrome")
else:
    print("Not a Palindrome")
```

Input:

Enter the string: step on no pets

Output:

Palindrome

DECIMAL TO BINARY IN PYTHON

How to convert the given input number from decimal to binary using two different methods. To find a binary equivalent, the decimal number has to be recursively divided by the value 2 until the decimal number reaches the value zero. After each and every division, the remainder has to be noted. The reverse of the remainder values gives the binary equivalent of the decimal number.

For example, the binary conversion of the decimal number 15 is shown in the below image.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

Binary equivalent of the given input decimal number

SAMPLE INPUT:

14

SAMPLE OUTPUT:

1110

Algorithm for converting Decimal to Binary

Step 1: Get an input value from the user

Step 2: If the input is greater than zero, divide the number by the value 2 and note the remainder

Step 3: Repeat Step 2 until the decimal number reaches zero

Step 4: Print the remainder values

Step 5: End

Method1: Using user-defined function

Python program to convert the given decimal number to binary using a user-defined function

```
def decimalToBinary(num):if num > 1:
    decimalToBinary(num // 2)
    print(num % 2, end = "")
number = int(input("Enter the decimal number: "))

#main() function
decimalToBinary(number)
```

Input:

Enter the decimal number: 25

Output:

11001

PREREQUISITE KNOWLEDGE: Built-In Functions in Python

Method 2: Using built-in function

When we use the built-in function, the output will be obtained with the prefix 'ob'.

Python program to convert the given decimal number to binary using a built-in function

```
number = int(input("Enter the decimal number: "))
print(bin(number))
```

Input:

Enter the decimal number: 56

Output:

Binary Number: 0b111000

ARMSTRONG NUMBER IN PYTHON

A number is said to be an Armstrong number when the sum of the nth power of each digit is equal to the number itself. Here, 'n' is the number of digits of the given number.

For example, an Armstrong number of three digits is an integer, such that the sum of the cubes of its digits is equal to the number itself.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

A single line saying "Armstrong Number", if the given input number is an Armstrong number.
Else, "Not an Armstrong Number".

SAMPLE INPUT:

473

SAMPLE OUTPUT:

Not an Armstrong number

PREREQUISITE KNOWLEDGE: Modules in Python

Method 1: Using math module

Algorithm to check whether the given number is Armstrong or not using the 'math' module

- Step 1:** Initialize result = 0 and temp = number
- Step 2:** Find the total number of digits in the number
- Step 3:** Repeat until (temp != 0)
- Step 4:** remainder = temp % 10
- Step 5:** result = result + pow(remainder,n)
- Step 6:** temp = temp / 10
- Step 7:** if (result == number)
- Step 8:** Print Armstrong Number
- Step 9:** Else
- Step 10:** Print Not an Armstrong Number
- Step 11:** End

Python program to check whether the given number is Armstrong or not using the 'math' module

```
#Python program to check whether the given number is Armstrong or not
from math import *
number = int(input("Enter the number : "))
result = 0
n = 0
temp = number;
while (temp != 0):
    temp =int(temp / 10)
    n = n + 1#Checking if the number is armstrong
```

```
temp = number
while (temp != 0):
    remainder = temp % 10
    result = result + pow(remainder, n)
    temp = int(temp/10)
if(result == number):
    print("Armstrong number")
else:
    print("Not an Armstrong number")
```

Input:

Enter the number: 407

Output:

Armstrong Number

Method 2: Using a while loop

Python program to check whether the given number is Armstrong or not using a while loop

```
#program to check whether the given number is Armstrong or not using a while loop
num = int(input("Enter the number: "))
sum = 0
temp = num
while (temp > 0):
    digit = temp % 10
    sum += digit ** 3
    temp //= 10
if (num == sum):
    print("Armstrong number")
else:
    print("Not an Armstrong number")
```

Input:

Enter the number: 925

Output:

Not an Armstrong Number

PYTHON PROGRAM TO PRINT ARMSTRONG NUMBERS WITHIN THE GIVEN RANGE IN PYTHON

```
#program to print Armstrong numbers within the given range in Python
lower = int(input("Enter lower range: "))
upper = int(input("Enter upper range: "))
for num in range(lower, upper + 1):
    order = len(str(num))
    sum = 0#find the sum of the cube of each digit
    temp = num
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
        temp //= 10
    if num == sum:
        print(num, end = ' ')
```

Input:

Enter lower range: 100

Enter upper range: 500

Output:

153 370 371 407

VOWEL OR CONSONANT IN PYTHON

To check whether the given character is vowel or consonant using different methods. Letters like a, e, i, o, u are called vowels. Rest all alphabets are called consonants.

INPUT FORMAT:

Input consists of a single character

OUTPUT FORMAT:

A single line saying "Vowel", if the given input character is a vowel. Else, "Consonant".

SAMPLE INPUT:

A

SAMPLE OUTPUT:

Vowel

Method 1: Using Built-In function

Algorithm to check whether the given character is vowel or consonant using the built-in function

Step 1: Get a character from the user

Step 2: Check whether the input is vowel or consonant by using the python built-in functions like(lower(), upper()).

Step 3: If the character is vowel print Vowel otherwise print Consonant

Step 4: End

Python program to check whether the given character is vowel or consonant using the built-in function

```
#program to check whether the given character is vowel or consonant using the built-in function
l = input("Enter the character: ")
if l.lower() in ('a', 'e', 'i', 'o', 'u'):
    print("Vowel")
elif l.upper() in ('A', 'E', 'I', 'O', 'U'):
    print("Vowel")
else:
    print("Consonant")
```

Input:

Enter the character: E

Output:

Vowel

PREREQUISITE KNOWLEDGE: Logical operators in Python

Method 2: Using Logical operator

Python program to check whether the given character is vowel or consonant using the logical operator

```
# Python Program to check whether the character is Vowel or Consonant using the logical operator
ch = input("Enter the character : ")
if(ch == 'a' or ch == 'e' or ch == 'i' or ch == 'o' or ch == 'u' or ch == 'A' or ch == 'E' or ch == 'I' or ch == 'O' or ch == 'U'):
    print("Vowel")
else:
    print("Consonant")
```

Input:

Enter the character: D

Output:

Consonant

Method 3: Using ASCII values

Python program to check whether the given character is vowel or consonant using the ASCII values

```
# Python Program to check whether the given character is Vowel or Consonant using ASCII values
ch = input("Enter the character : ")
if(ord(ch) == 65 or ord(ch) == 69 or ord(ch) == 73 or ord(ch) == 79 or ord(ch) == 85 or ord(ch) == 97 or ord(ch) == 101 or ord(ch) == 105 or ord(ch) == 111 or ord(ch) == 117):
    print("Vowel")
else:
    print("Consonant")
```

Input:

Enter the character: v

Output:

Consonant

GREATEST OF TWO NUMBERS IN PYTHON

How to print the greatest of two numbers using different methods.

INPUT FORMAT:

Input Consists of 2 integers

OUTPUT FORMAT:

A single integer which is the greatest of the two given input integers

SAMPLE INPUT:

45
86

SAMPLE OUTPUT:

86

PREREQUISITE KNOWLEDGE: Built-In Functions in Python

Method 1: Using Built-In Function

Algorithm to print the greatest of two numbers using the built-in function

- Step 1:** Get 2 inputs from the user
Step 2: Find the greatest of two numbers using the built-in function 'max'
Step 3: Print the output
Step 5: End

Python program to print the greatest of two numbers using the built-in function

```
# Python program to find the greatest of two numbers using the built-in function
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
print(max(num1, num2), "is greater")
```

Input:

Enter the first number: 23
Enter the second number: 45

Output:

45 is greater

Method 2: Using if-else statements

Algorithm to print the greatest of two numbers using if-else statements

- Step1:** Get the 2 inputs from the user
Step 2: Check whether the first value is greater than the second value
Step 3: If yes, print the first value
Step 4: Else, print the second value
Step 5: End

Python program to print the greatest of two numbers using if-else statements

```
# Python Program to find Largest of two Numbers using if-else statements
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
if(a >= b):
    print(a, "is greater")
else:
    print(b, "is greater")
```

Input:

Enter the first number: 50
Enter the second number: 45

Output:

50 is greater

Method 3: Using Arithmetic operator

Python program to print the greatest of two numbers using an arithmetic operator

```
# Python Program to find the largest of two numbers using an arithmetic operator
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
if(a - b > 0):
    print(a, "is greater")
else:
    print(b, "is greater")
```

Input:

Enter the first number: 90

Enter the second number: 100

Output:

100 is greater

PASCAL TRIANGLE IN PYTHON

Pascal's triangle is a triangular array of numbers in which the values at the ends of the rows are 1 and each of the others is the sum of the nearest two numbers in the row above. Below is a pictorial representation of Pascal's triangle having 4 rows.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

Refer sample output

SAMPLE INPUT:

3

SAMPLE OUTPUT:

```
1
1 1
```



```
1 2 1
```

Algorithm to Print Pascal's Triangle in Python

Step 1: Get an input value from the user

Step 2: Using a for loop which ranges from 0 to n-1, append the sub-lists into the list

Step 3: Then append 1 into the sub-lists

Step 4: Then use a for loop to determine the value of the number inside the triangle

Step 5: Print Pascal's triangle according to the format

Step 6: Exit

Python program for Printing the Pascal's triangle in Python

```
# Python program to print pascal's triangle
n = int(input("Enter the number: "))
a = [] #an empty list
for i in range(n):
    a.append([])
    a[i].append(1)
    for j in range(1, i):
        a[i].append(a[i - 1][j - 1] + a[i - 1][j])
    if(n != 0):
        a[i].append(1)
for i in range(n):
    print(" " * (n - i), end = " ", sep = " ")
    for j in range(0, i + 1):
        print('{0:6}'.format(a[i][j]), end = " ", sep = " ")
    print()
```

Input:

Enter the number: 4

Output:

```
1
1 1
1 2 1
1 3 3 1
```

PYRAMID PATTERN IN PYTHON

How to print different half Pyramid Patterns using for loop in Python.

First, let us see how to print half pyramid pattern using asterisk symbol.

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

Refer sample output

SAMPLE INPUT:

3

SAMPLE OUTPUT:

```
*  
* *  
* * *
```

Algorithm for printing Half Pyramid Pattern in Python

Step 1: Get the number of rows from the user

Step 2: Iterate through the given number using outer for loop to handle the number of rows

Step 3: Iterate through the inner loop to handle the number of columns. Inner loop iteration depends on the values of the outer loop

Step 4: Print asterisk or number in pyramid or diamond pattern using the print() function

Step 5: Add a new line after each row (after each iteration of outer for loop) to display the pattern appropriately

Python program for printing Half Pyramid pattern using asterisk symbol

```
#Printing pyramid pattern using the asterisk symbol  
size = int(input("Enter the number of rows: "))  
m = (2 * size) - 2  
for i in range(0, size):  
    for j in range(0, m):  
        print(end = " ")  
    m = m - 1 # decrementing m after each loop  
    for j in range(0, i + 1):  
        print("*", end = ' ')  
    print()
```

Input:

Enter the number of rows: 4

Output:

```
*  
* *  
* * *  
* * * *
```

Next, let us see how to print half pyramid pattern using numbers.

Half pyramid pattern using number - 1:

INPUT FORMAT:

Input consists of an integer

OUTPUT FORMAT:

Refer sample output

SAMPLE INPUT:

```
4
```

SAMPLE OUTPUT:

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

Python program for printing Half Pyramid pattern using numbers

```
#Python program to print the number pattern  
n = int(input("Enter the number of rows: "))  
for num in range(1, n + 1):  
    for i in range(num):  
        print (num, end = " ") #printing number  
    print()
```

Input:

Enter the number of rows: 3

Output:

```
1  
2 2  
3 3 3
```

Half pyramid pattern using number - 2:

SAMPLE INPUT:

4

SAMPLE OUTPUT:

1
1 2
1 2 3
1 2 3 4

Python program for printing Half Pyramid pattern using numbers

```
# Python program to print pyramid number pattern
num = int(input("Enter the number of rows: "))
for row in range(1, num + 1):
    for column in range(1, row + 1):
        print(column, end = ' ')
    print("")
```

Input:

Enter the number of rows: 3

Output:

1
1 2
1 2 3

Half pyramid pattern using number - 3:

SAMPLE INPUT:

4

SAMPLE OUTPUT:

1
2 1
3 2 1
4 3 2 1

Python program for printing Half Pyramid pattern using numbers

```
# Python program to print pyramid number pattern
num = int(input("Enter the number of rows: "))
for row in range(1, num + 1):
```

```
for column in range(row, 0, -1):
    print(column, end = ' ')
print("")
```

Input:

Enter the number of rows: 3

Output:

```
1
2 1
3 2 1
```

DIAMOND PATTERN IN PYTHON

INPUT FORMAT:

5

OUTPUT FORMAT:

Refer sample output

SAMPLE INPUT:

3

SAMPLE OUTPUT:

```
  *
 * * *
* * * * *
 * * *
  *
```

Python program for printing the full diamond pattern using stars

```
#top half
rows = int(input("Enter the number of rows:"))
k = 0
for i in range(1, rows + 1):
    # print spaces
    for j in range(1, (rows - i) + 1):
        print(end = " ")

    # let's print stars
    while k != (2 * i - 1):
        print("*", end = "")
        k = k + 1
```

```

k = 0# add a line break
print()

#bottom half
k = 2
m = 1for i in range(1, rows):
    # print spacesfor j in range (1, k):
        print(end = " ")
    k = k + 1# print starwhile m <= (2 * (rows - i) - 1):
        print("*", end = "")
    m = m + 1
    m = 1#add a line break
    print()

```

Next, let us discuss how to print the right half diamond pattern using stars.

SAMPLE INPUT:

```
4
```

SAMPLE OUTPUT:

```

*
* *
* * *
* * * *
* * *
* * *
* *
*

```

Python Program to print the right half diamond pattern using stars

```

#Python Program to print full right angle triangle pattern
rows = input("Enter the number of rows: ")
rows = int (rows)
for i in range (0, rows):
    for j in range(0, i + 1):
        print("*", end = ' ')
    print()

for i in range (rows, 0, -1):

```

```
for j in range(0, i -1):  
    print("*", end = ' ')  
print()
```

Input:

Enter the number of rows: 3

Output:

```
*  
* *  
* * *  
* *  
*
```

MATRIX ADDITION IN PYTHON

let us discuss how to add values given in the two input matrices.

SAMPLE INPUT:

```
mat1 = [[4, 3], [5, 4]]  
mat2 = [[1, 2], [3, 6]]
```

SAMPLE OUTPUT:

```
Addition of two matrices  
5 5 8 10
```

Algorithm to perform Matrix Addition

Step 1: Input the matrix 1 elements

Step 2: Input the matrix 2 elements

Step 3: Repeat from i = 0 to number of rows

Step 4: Repeat from j = 0 to number of columns

Step 5: $\text{mat3}[i][j] = \text{mat1}[i][j] + \text{mat2}[i][j]$

Step 6: Print mat3

Step 7: End

Python Program to Perform Matrix Addition

```
# Python program to perform matrix addition  
mat1 = [[1, 2], [3, 4]] #the first matrix  
mat2 = [[1, 2], [3, 4]] #the second matrix  
mat3 = [[0, 0], [0, 0]] #an empty matrix to store the result#
```

```

adding two matrices
for i in range(0, 2):
    for j in range(0, 2):
        mat3[i][j] = mat1[i][j] + mat2[i][j]

#printing the resultant matrix
print("Addition of two matrices")
for i in range(0, 2):
    for j in range(0, 2):
        print(mat3[i][j], end = " ")
    print()

```

Output:

Addition of two matrices
 2 4
 6 8

MATRIX MULTIPLICATION IN PYTHON

SAMPLE INPUT:

```

mat1 = [[1, 2], [3, 4]]
mat2 = [[1, 2], [3, 4]]

```

SAMPLE OUTPUT:

```

[7, 10]
[15, 22]

```

Algorithm to perform matrix multiplication

Step 1: Input matrix 1 elements

Step 2: Input matrix 2 elements

Step 3: Repeat from i = 0 to number of rows of the first matrix

Step 4: Repeat from j = 0 to number of columns of the second matrix

Step 5: Repeat from k = 0 to number of rows of the second matrix

Step 6: $\text{mat3}[i][j] += \text{mat1}[i][k] * \text{mat2}[k][j]$

Step 7: Print mat3

Step 8: End

Python program to perform matrix multiplication in Python

```

# 2x2 matrix

```



```

X = [[10, 9], [8, 6]]
# 2x2 matrix
Y = [[1, 2], [3, 4]]
# result is 2x2
result = [[0, 0], [0, 0]]
# iterate through rows of X
for i in range(len(X)):
    # iterate through columns of Y
    for j in range(len(Y[0])):
        # iterate through rows of Y
        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]
print("Multiplication of two matrices")
for r in result:
    print(r)

```

Output:

Multiplication of two matrices

[37, 56]

[26, 40]

TRANSPOSE OF A MATRIX IN PYTHON

The transpose of a matrix is a new matrix whose rows are the columns of the original matrix and vice versa.

INPUT FORMAT:

Input consists of a matrix

OUTPUT FORMAT:

Transpose of the given input matrix

SAMPLE INPUT:

```

1 2 3
4 5 6
7 8 9

```

SAMPLE OUTPUT:

```

1 4 7
2 5 8
3 6 9

```

Algorithm to print the transpose of a matrix

Step 1: Create an empty input matrix to store user input matrix elements

Step 2: Input number of rows and number of columns from the user

Step 3: Input row and column elements from the user

Step 4: Append the user input row and column elements into an empty matrix

Step 5: Create an empty transpose matrix to store the output

Step 6: Append row-wise elements of the input matrix to the column of the empty transpose matrix

Step 7: Append column-wise elements of the input matrix to the row of the transpose matrix

Step 8: Print transpose matrix

Step 9: End

Python program to print the transpose of a matrix

```
matrix = [] #an empty matrix to store user input elements
row = int(input("Enter number of rows: "))
column = int(input("Enter number of columns: "))
for i in range(row):
    matrix.append([])
    #getting row and column elements from the user and appending them to the empty matrix
    for j in range(column):
        num = int(input("Enter the element: "))
        matrix[i].append(num)

#printing the input matrix
print("Input matrix: ")
for i in range (row):
    for j in range (column):
        print(matrix[i][j], end = " ")
    print()

#interchanging row elements as column elements and vice versa
transpose = []
for j in range(column):
    transpose.append([])
    for i in range (row):
        t_num = matrix[i][j]
        transpose[j].append(t_num)

#printing the transpose matrix
print("Transpose matrix: ")
for i in range (row):
    for j in range (column):
        print (transpose[i][j], end = ' ')
    print()
```

Input:

Enter number of rows: 2
Enter number of columns: 2
Enter the element: 3
Enter the element: 5
Enter the element: 7
Enter the element: 9

Output:

Input matrix:

3 5

7 9

Transpose matrix:

3 7

5 9

SUM OF ARRAY IN PYTHON

Given the number of elements of the array and the array elements. Output - sum of all elements of the array.

INPUT FORMAT:

Input the size of the array and array elements

OUTPUT FORMAT:

Sum of all array elements

SAMPLE INPUT:

3
1
2
3

SAMPLE OUTPUT:

6

Algorithm to find the sum of elements of the given array

Step 1: Get the size of the array from the user

Step 2: Initialize an empty list `lst = []`

Step 3: Get array elements from the user using a for loop

Step 4: Using the for loop, append each array elements to the list

Step 5: Using the built-in function `sum()`, find the sum of all the elements of the list

Step 6: Print the result

Step 7: End

Python program to find the sum of elements of the given array

```
#Python program to add all the array elements using the built-in function
lst = []
num = int(input("Enter the size of the array: "))
print("Enter array elements: ")
for n in range(num):
    numbers = int(input())
    lst.append(numbers)
print("Sum:", sum(lst))
```

Input:

Enter the size of the array: 4

Enter array elements:

3

5

7

8

Output:

Sum: 23

THE SMALLEST ELEMENT OF AN ARRAY IN PYTHON

INPUT FORMAT:

Input the size of the array and the array elements

OUTPUT FORMAT:

The smallest of all the array elements

SAMPLE INPUT:

3

3

6

8

SAMPLE OUTPUT:

3

Method 1: Using for loop

Algorithm to find the smallest element of the given array using for loop

- Step 1:** Input the array elements
- Step 2:** Initialize small = list[0]
- Step 3:** Repeat from n= 0 to the size of the array
- Step 4:** if(list[n] < small)
- Step 5:** small = list[n]
- Step 6:** Print small element
- Step 7:** End

Python program to print the smallest element of the given array using for loop

```
#Python program to find the smallest element of the array using 'for' loop
lst = []
num = int(input("Enter the size of the array: "))
print("Enter array elements: ")
for n in range(num):
    numbers = int(input())
    lst.append(numbers)
small = lst[0]
for n in range(num):
    if(lst[n] < small):
        small = lst[n]
print("The largest element of the given list is:", small)
```

Method 2: Using the built-in function

Python program to print the smallest element of the given array using the built-in function

```
#Python program to find the smallest element of the array using the built-in function
lst = []
num = int(input("Enter the size of the array: "))
print("Enter the array elements: ")
for n in range(num):
    numbers = int(input())
    lst.append(numbers)
print("The smallest element of the given list is:", min(lst))
```

Input:

Enter the size of the array: 4

Enter array elements:

5

7

2

8

Output:

The smallest element of the given list is: 2

THE LARGEST ELEMENT OF AN ARRAY IN PYTHON

INPUT FORMAT:

Input the size of the array and the array elements

OUTPUT FORMAT:

The largest of all the array elements

SAMPLE INPUT:

3

3

6

8

SAMPLE OUTPUT:

8

Method 1: Using for loop

Algorithm to find the largest element of the given array using for loop

Step 1: Input the array elements

Step 2: Initialize large = list[0]

Step 3: Repeat from n = 0 to the size of the array

Step 4: if(list[n] > large)

Step 5: large = list[n]

Step 6: Print large element

Step 7: End

Python program to find the largest element of the given array using for loop

```
#Python program to find the largest element of the array using 'for' loop
lst = []
num = int(input("Enter the size of the array: "))
print("Enter array elements: ")
for n in range(num):
    numbers = int(input())
    lst.append(numbers)
large = lst[0]
for n in range(num):
    if(lst[n] > large):
        large = lst[n]
print("The largest element of the given list is:", large)
```

Method 2: Using built-in function

Python program to find the largest element of the given array using the built-in function

```
#Python program to find the largest element of the array using the built-in function
lst = []
num = int(input("Enter the size of the array: "))
print("Enter array elements: ")
for n in range(num):
    numbers = int(input())
    lst.append(numbers)
print("The largest element of the given list is:", max(lst))
```

Input:

Enter the size of the array: 4

Enter array elements:

5

7

2

8

Output:

The smallest element of the given list is: 8

FINDING STRING LENGTH IN PYTHON

How to find the length of the given input string. In Python, special characters like newlines, tabs, whitespaces, escape characters can be added to the length of the string. Let us discuss the different ways of calculating the length of the given input string.

INPUT FORMAT:

Input consists of a string

OUTPUT FORMAT:

Length of the given input string

SAMPLE INPUT:

Hello

SAMPLE OUTPUT:

5

Method 1: Using for loop

Algorithm to find the length of a string using for loop

Step 1: Input the string from the user

Step 2: Initialize counter = 0

Step 3: Using a 'for' loop, traverse through the length of the string

Step 4: Increment the variable 'counter' for each character of the string

Step 5: Print the counter variable

Step 6: End

Python program to find the length of a string using for loop

```
# Python program to find the length of the string using 'for' loop
def findLen(str):
    counter = 0
    for i in str:
        counter += 1
    return counter

# main() method
str = input("Enter the string: ")
print("Length of the string:", findLen(str))
```

Input:

Enter the string: FACE Prep

Output:

Length of the string: 9

Method 2: Using len() method

Python program to find the length of a string using len() method

```
#Python program to print the length of the string using 'len' method
string = input("Enter the string: ")
print("Length of the string:", len(string))
```

Input:

Enter the string: Hello World

Output:

Length of the string: 11

Next, let us see how to find the length of different strings stored in a list.

Python program to find the length of strings stored in a list

For this example, a list of string items is created. 'for' loop is used to iterate through the list items. In each iteration, the length of the current item (string) is displayed. Have a look at the code and output.

```
# Python program to find the length of a list of strings
List_string_len = ["FACE Prep", "Python", "Programs"]
for str_len in List_string_len:
    print(str_len, "=", len(str_len))
```

Output:

FACE Prep = 9

Python = 6

Programs = 8

STRING SLICING IN PYTHON

Slicing:

Slicing is done in a string to obtain sub-strings of the original string. Python allows us to specify where to start slicing, and where to end. Also, we can specify the step value as shown in the below syntax.

Syntax:

```
string_object[start : stop : steps]
```

Here, slicing starts from the index 'start' and ends at the index 'stop' in the step of 'steps'. The default value for

start - 0

stop - the last index of the string and

step - 1

string_object[: stop] will slice the given string from starting, until 'stop' index and

string_object[start:] will slice the given string from 'start' index until the end of the string.

INPUT FORMAT:

Input consists of a string

OUTPUT STRING:

The output contains sliced sub-string

SAMPLE INPUT:

```
Python[2:5]
```

SAMPLE OUTPUT:

```
tho
```

Note: 'Stop' index is exclusive

PREREQUISITE KNOWLEDGE: Lists in Python

Algorithm to slice the given input string

We clearly don't need an algorithm for slicing a string in Python. It can be done using a simple slicing operator i.e, using colon.

Method 1: Using positive indexing

Python program to slice the given string using positive indexing

```
#Python program to slice the string "FACE Prep" using positive indexing  
str1 = "FACE Prep"#printing the sliced sub-string from the index 0 to 3
```

```
print(str1[0:4])  
#printing the sliced sub-string from the index 0 to 9 in steps of 2  
print(str1[0:9:2])
```

Output:

FACE
FC rp

Method 2: Using negative indexing

Under the concept of negative indexing, the index value of the last character of the string is always -1. Using negative indexing, we can easily retrieve the last character of the string without knowing its entire length.

Python program to slice the given string using negative indexing

```
#Python program to slice the string "FACE Prep" using negative indexing  
str1 = "FACE Prep"#printing the last character  
print(str1[-1])  
#printing the second last character  
print(str1[-2])  
#printing the sliced sub-string from the index -4 to -3  
print(str1[-4:-2])  
#printing the sliced sub-string from the index -10 to -6  
print(str1[-10:-5])
```

Output:

p
e
Pr
FACE

STRING CONCATENATION IN PYTHON**INPUT FORMAT:**

Input consists of two strings

OUTPUT FORMAT:

The output contains merged or joined strings

SAMPLE INPUT:

Hello

World

SAMPLE OUTPUT:

HelloWorld

Algorithm to concatenate two strings

We clearly don't need an algorithm for concatenating two strings in Python. It can be done using a simple arithmetic operator i.e, addition operator.

Python program to concatenate two strings

```
#Python program to concatenate two strings
str1 = input("Enter the first string: ") #input string1
str2 = input("Enter the second string: ") #input string2
print("Concatenated string:", str1 + str2) #prints the concatenated string
```

Input:

Enter the first string: PACE

Enter the second string: College

Output:

Concatenated string: PACECollege

STRING REVERSAL IN PYTHON

However, strings can be reversed in several different ways. Let's discuss the algorithm and solution for that.

INPUT FORMAT:

Input consists of a string

OUTPUT FORMAT:

A reversed string of the original string

SAMPLE INPUT:

Hello

SAMPLE OUTPUT:

olleH

Method 1: Using a slice operator

Algorithm to reverse the string using the slice operator

- Step 1:** Get an input string from the user
- Step 2:** Using the slice operator reverse the string
- Step 3:** Print the reversed string
- Step 4:** End

Python program to reverse the string using the slice operator

```
#Python program to reverse the string using a slice operator
a = input("Enter the string: ")
print("Reversed string:", a[::-1])
```

Input:

Enter the string: FACE

Output:

Reversed string: ECAF

Method 2: Using a while loop

Python program to reverse the string using a while loop

```
#Python program to reverse the string using a while loop
str1 = input("Enter the string: ") #initial string
reversedString = ""
index = len(str1) #calculating length of the string
while(index > 0):
    reversedString += str1[index - 1 ]
    index = index - 1
print("Reversed string:", reversedString) #reversed string
```

Input:

Enter the string: Prep

Output:

Reversed string: perP

CHECK FOR SUBSTRING IN PYTHON

In Python, there are different ways to achieve this.

1. Using 'in' operator
2. Using the method 'find()'
3. Using the method 'contains()'

INPUT FORMAT:

Input consists of 2 strings

OUTPUT FORMAT:

A single line saying "Substring is present" if the given string contains the specified substring. Else, "Substring is not present".

SAMPLE INPUT:

Enter the string: Welcome to Python Programming
Enter the substring: Python

SAMPLE OUTPUT:

Substring is present

PREREQUISITE KNOWLEDGE: Operators in Python

Method 1: Check for the substring in the string using 'in' operator

Python program to check for the substring in the string using 'in' operator

```
# Python program to check for the substring in the given string using 'in' operator
str1 = input("Enter the string: ")
substr1 = input("Enter the substring: ")
# checking whether the string 'str1' contains "FACE Prep"if(substr1 in str1):
    print ("Substring is present")
else:
    print ("Substring is not present")
```

Method 2: Check for the substring in the string using the method 'find()'

Python program to check for the substring in the string using the method 'find()'

```
# Python program to check for the substring in the given string using the method 'find()'
str1 = input("Enter the string: ")
substr1 = input("Enter the substring: ")
res = str1.find(substr1)
if(res >= 0):
    print ("Substring is present")
else:
    print ("Substring is not present")
```

Method 3: Check for the substring in the string using the method contains()

Python program to check for the substring in the string using the method 'contains()'

```
# Python program to check for the substring in the string using the method 'contains()' from
operator import *
str1 = input("Enter the string: ")
substr1 = input("Enter the substring: ")
if(contains(str1, substr1)):
    print("Substring is present")
else:
    print ("Substring is not present")
```

Input:

Enter the string: FACE Prep

Enter the substring: Prep

Output:

Substring is present

REVERSING EACH WORD OF A STRING IN PYTHON

Steps to reverse each word of a string

First, split the entire string into a list of words. Then reverse each word and store the reversed words in a new list. At last, join the new list of words to create a new string.

INPUT FORMAT:

Input consists of a string

OUTPUT FORMAT:

Reversed string

SAMPLE INPUT:

Python program

SAMPLE OUTPUT:

nohtyP margorp

PREREQUISITE KNOWLEDGE: Strings in Python

Algorithm to reverse each word of a string in Python

Step 1: Input a string and store it in a variable s

Step 2: Then split the string into a list of words using the built-in function 'split()'

Step 3: Reverse each word and store them in a new list nw

Step 4: Join the new list of words and make a new string ns

Python program to reverse each word of a string

```
# Python program to Reverse each word of a string# function definition
def reverseword(s):
    w = s.split(" ")
    # Splitting the string into a list of words# reversing each word and creating a new list of words
    nw = [i[::-1] for i in w]
    # Joining the new list of words to form a new string
    ns = " ".join(nw)
    return ns

# main() method
s = input("Enter the string: ")
print(reverseword(s))
```

Input:

Enter the string: intelligence

Output:

ecnegilletni

REMOVING A CHARACTER FROM STRING IN PYTHON

INPUT FORMAT:

Input consists of a string

OUTPUT FORMAT:

A string with one character removed

SAMPLE INPUT:

```
HelloWorld!
```

SAMPLE OUTPUT:

```
HelloWorld
```

PREREQUISITE KNOWLEDGE: Strings in Python

Method 1: Using the method 'replace()'

We can use the method 'replace()' to replace a character with a new character. This function accepts two arguments. The first argument is the character that has to be removed from the string. Second, an empty string. If we provide an empty string as the second argument to this function, then the particular character will get replaced with space which is an indirect way of removing the character.

In Python, the string is immutable in Python. So, this function will return a new string with characters removed. The original string will remain unchanged.

Python program to remove a character from the string using the method 'replace()'

```
#Python program to remove a character from the string using the method 'replace()'
s = "You are a Python Developer"
print(s.replace('e', ""))
```

Output:

You ar a Python Dvlopr

Method 2: Using a slice operator

Python program to remove a character from the string using a slice operator

```
#Python program to remove a character from the string using a slice operator
def remove(string, i):# Characters before the ith indexed are stored in a variable 'a'
    a = string[: i]

    # Characters after the ith indexed are stored in a variable 'b'
    b = string[i + 1: ]

    # Returning string after removing ith indexed characterreturn a + b
# main() method
string = input("Enter the string: ")
i = int(input("Enter the index from which a character has to be removed: "))
# Printing the new string
print(remove(string, i))
```

Input:

Enter the string: FACE Prep

Enter the index from which a character has to be removed: 3

Output:

FAC Prep

REMOVING DUPLICATES FROM A STRING

INPUT FORMAT:

Input consists of a string

OUTPUT FORMAT:

The output consists of a string with unique characters

SAMPLE INPUT:

FACEFACE

SAMPLE OUTPUT:

FACE

PREREQUISITE KNOWLEDGE: Strings in Python

Algorithm to remove duplicate characters from the given input string

Step 1: Input the string from the user

Step 2: Traverse through the length of the string

Step 3: Check if the characters are repeating

Step 4: If yes, break

Step 5: Else, print the characters

Python program to remove duplicate characters from the given input string

```
# Python program to remove duplicate characters from the given string
def removeDuplicate(str1, n):
    # Used as an index in the modified string
    index = 0# Traversing through all characters
    for i in range(0, n):# Checking if a character is repeated
        for j in range(0, i + 1):
            if (str1[i] == str1[j]):break# If not present, then add it to result
        if (j == i):
            str1[index] = str1[i]
            index += 1return "".join(str1[:index])

# main() method
str1 = input("Enter the string: ")
n = len(str1)
```

```
print(removeDuplicate(list(str1), n))
```

Input:

Enter the string: programming

Output:

progamin