# Python program to check if a number is positive, negative or zero

In this example, you will learn to check whether a number entered by the user is positive, negative or zero. This problem is solved using if...elif...else and nested if...else statement.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)
* [Python Input, Output and Import](https://www.programiz.com/python-programming/input-output-import)

## Source Code: Using if...elif...else

[Python program to check if a number is positive, negative or zero](https://www.programiz.com/python-programming/examples/positive-negative-zero)

num = float(input("Enter a number: "))

if num > 0:

print("Positive number")

elif num == 0:

print("Zero")

else:

print("Negative number")

The output of both programs will be same.

Output 1

Enter a number: 2

Positive number

Output 2

Enter a number: 0

Zero

A number is positive if it is greater than zero. We check this in the expression of if. If it is False, the number will either be zero or negative. This is also tested in subsequent expression.

# Python Program to Check if a Number is Odd or Even

In this example, you will learn to check whether a number entered by the user is even or odd.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python Operators](https://www.programiz.com/python-programming/operators)
* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)

A number is even if it is perfectly divisible by 2. When the number is divided by 2, we use the remainder operator % to compute the remainder. If the remainder is not zero, the number is odd.

# Python program to check if the input number is odd or even.

# A number is even if division by 2 give a remainder of 0.

# If remainder is 1, it is odd number.

num = int(input("Enter a number: "))

if (num % 2) == 0:

print("{0} is Even".format(num))

else:

print("{0} is Odd".format(num))

## Basic formatting

Simple positional formatting is probably the most common use-case. Use it if the order of your arguments is not likely to change and you only have very few elements you want to concatenate.

Since the elements are not represented by something as descriptive as a name this simple style should only be used to format a relatively small number of elements.

### Old

'%s %s' % ('one', 'two')

### New

'{} {}'.format('one', 'two')

### Output

*one* *two*

### Old

'%d %d' % (1, 2)

### New

'{} {}'.format(1, 2)

### Output

*1* *2*

With new style formatting it is possible (and in Python 2.6 even mandatory) to give placeholders an explicit positional index.

This allows for re-arranging the order of display without changing the arguments.

This operation is not available with old-style formatting.

### New

'{1} {0}'.format('one', 'two')

### Output

*two* *one*

# Python Program to Check Leap Year

In this program, you will learn to check whether a year is leap year or not. We will use nested if...else to solve this problem.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python Operators](https://www.programiz.com/python-programming/operators)
* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)

A leap year is exactly divisible by 4 except for century years (years ending with 00). The century year is a leap year only if it is perfectly divisible by 400. For example,

2017 is not a leap year

1900 is a not leap year

2012 is a leap year

2000 is a leap year

## Source Code

# Python program to check if the input year is a leap year or not

year = 2000

# To get year (integer input) from the user

# year = int(input("Enter a year: "))

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))

Output

2000 is a leap year

You can change the value of year in the source code and run it again to test this program.

# Python Program to Swap Two Variables

In this example, you will learn to swap two variables by using a temporary variable and, without using temporary variable.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python Variables and Data Types](https://www.programiz.com/python-programming/variables-datatypes)
* [Python Input, Output and Import](https://www.programiz.com/python-programming/input-output-import)
* [Python Operators](https://www.programiz.com/python-programming/operators)

## Source Code: Using temporary variable

* [script.py](https://www.programiz.com/python-programming/examples/swap-variables)
* [IPython Shell](https://www.programiz.com/python-programming/examples/swap-variables)

# Python program to swap two variables

# To take input from the user

# x = input('Enter value of x: ')

# y = input('Enter value of y: ')

x = 5

y = 10

# create a temporary variable and swap the values

temp = x

x = y

y = temp

print('The value of x after swapping: {}'.format(x))

print('The value of y after swapping: {}'.format(y))

Output

The value of x after swapping: 10

The value of y after swapping: 5

In this program, we use the temp variable to temporarily hold the value of x. We then put the value of y in x and later temp in y. In this way, the values get exchanged.

# Python loop examples:

# Python Program to Check Prime Number

Example to check whether an integer is a prime number or not using for loop and if...else statement. If the number is not prime, it's explained in output why it is not a prime number.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)
* [Python for Loop](https://www.programiz.com/python-programming/for-loop)
* [Python break and continue](https://www.programiz.com/python-programming/break-continue)

A positive integer greater than 1 which has no other factors except 1 and the number itself is called a prime number. 2, 3, 5, 7 etc. are prime numbers as they do not have any other factors. But 6 is not prime (it is composite) since, 2 x 3 = 6.

## Source Code

* [script.py](https://www.programiz.com/python-programming/examples/prime-number)
* [IPython Shell](https://www.programiz.com/python-programming/examples/prime-number)

# Python program to check if the input number is prime or not

num = 407

# take input from the user

# num = int(input("Enter a number: "))

# prime numbers are greater than 1

if num > 1:

# check for factors

for i in range(2,num):

if (num % i) == 0:

print(num,"is not a prime number")

print(i,"times",num//i,"is",num)

break

else:

print(num,"is a prime number")

# if input number is less than

# or equal to 1, it is not prime

else:

print(num,"is not a prime number")

407 is not a prime number

11 times 37 is 407

In this program, variable num is checked if it's prime or not. Numbers less than or equal to 1 are not prime numbers. Hence, we only proceed if the num is greater than 1.

We check if num is exactly divisible by any number from 2 to num - 1. If we find a factor in that range, the number is not prime. Else the number is prime.

We can decrease the range of numbers where we look for factors.

In the above program, our search range is from 2 to num - 1.

We could have used the range, [2, num / 2] or [2, num \*\* 0.5]. The later range is based on the fact that a composite number must have a factor less than square root of that number; otherwise the number is prime.

You can change the value of variable num in the above source code and test for other integers (if you want).

# Python Program to Find the Factorial of a Number

In this article, you'll learn to find the factorial of a number and display it.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)
* [Python for Loop](https://www.programiz.com/python-programming/for-loop)

The factorial of a number is the product of all the integers from 1 to that number.

For example, the factorial of 6 (denoted as 6!) is 1\*2\*3\*4\*5\*6 = 720. Factorial is not defined for negative numbers and the factorial of zero is one, 0! = 1.

## Source Code

# Python program to find the factorial of a number provided by the user.

# change the value for a different result

num = 7

# uncomment to take input from the user

#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero

if num < 0:

print("Sorry, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

else:

for i in range(1,num + 1):

factorial = factorial\*i

print("The factorial of",num,"is",factorial)

Output

The factorial of 7 is 5040

Note: To test the program, change the value of num. Try negative numbers as well.

Here, the number whose factorial is to be found is stored in num and we check if the number is negative, zero or positive using if...elif...else statement. If the number is positive, we use for loop and range() function to calculate the factorial.

# Python Program to Print all Prime Numbers in an Interval

In this program, you'll learn to print all prime numbers within an interval using for loops and display it.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)
* [Python for Loop](https://www.programiz.com/python-programming/for-loop)
* [Python break and continue](https://www.programiz.com/python-programming/break-continue)

A positive integer greater than 1 which has no other factors except 1 and the number itself is called a prime number.

2, 3, 5, 7 etc. are prime numbers as they do not have any other factors. But 6 is not prime (it is composite) since, 2 x 3 = 6.

# Python program to display all the prime numbers within an interval

# change the values of lower and upper for a different result

lower = 900

upper = 1000

# uncomment the following lines to take input from the user

#lower = int(input("Enter lower range: "))

#upper = int(input("Enter upper range: "))

print("Prime numbers between",lower,"and",upper,"are:")

for num in range(lower,upper + 1):

# prime numbers are greater than 1

if num > 1:

for i in range(2,num):

if (num % i) == 0:

break

else:

print(num)

Output

Prime numbers between 900 and 1000 are:

907

911

919

929

937

941

947

953

967

971

977

983

991

997

Here, we store the interval as lower for lower interval and upper for upper interval, and find prime numbers in that range. Visit this page to understand the [code to check for prime numbers](http://www.programiz.com/python-programming/examples/prime-number)

**Python while loop**

# Python Program to Print the Fibonacci sequence

In this program, you'll learn to print the Fibonacci sequence using while loop.

**To understand this example, you should have the knowledge of following** [**Python programming**](https://www.programiz.com/python-programming) **topics:**

* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)
* [Python while Loop](https://www.programiz.com/python-programming/while-loop)

A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8....

The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means to say the nth term is the sum of (n-1)th and (n-2)th term.

# Program to display the Fibonacci sequence up to n-th term where n is provided by the user

# change this value for a different result

nterms = 10

# uncomment to take input from the user

#nterms = int(input("How many terms? "))

# first two terms

n1 = 0

n2 = 1

count = 2

# check if the number of terms is valid

if nterms <= 0:

print("Please enter a positive integer")

elif nterms == 1:

print("Fibonacci sequence upto",nterms,":")

print(n1)

else:

print("Fibonacci sequence upto",nterms,":")

print(n1,",",n2,end=', ')

while count < nterms:

nth = n1 + n2

print(nth,end=' , ')

# update values

n1 = n2

n2 = nth

count += 1

Output

Fibonacci sequence upto 10 :

0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

Note: To test this program, change the value of nterms.

Here, we store the number of terms in nterms. We initialize the first term to 0 and the second term to 1.

If the number of terms is more than 2, we use a while loop to find the next term in the sequence by adding the preceding two terms. We then interchange the variables (update it) and continue on with the process.

# Python Program to Check Armstrong Number

In this example, you will learn to check whether an n-digit integer is an Armstrong number or not.

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. For example, **371** is an Armstrong number since 3\*\*3 + **7**\*\*3 + 1\*\*3 = **371**. Write a program to find all Armstrong number in the range of 0 and **999**.

To understand this example, you should have the knowledge of following [Python programming](https://www.programiz.com/python-programming) topics:

* [Python if...else Statement](https://www.programiz.com/python-programming/if-elif-else)
* [Python while Loop](https://www.programiz.com/python-programming/while-loop)

A positive integer is called an Armstrong number of order n if

abcd... = an + bn + cn + dn + ...

In case of an Armstrong number of 3 digits, the sum of cubes of each digits is equal to the number itself. For example:

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3 // 153 is an Armstrong number.

## Source Code: Check Armstrong number (for 3 digits)

# Python program to check if the number provided by the user is an Armstrong number or not

# take input from the user

# num = int(input("Enter a number: "))

# initialize sum

sum = 0

# find the sum of the cube of each digit

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* 3

temp //= 10

# display the result

if num == sum:

print(num,"is an Armstrong number")

else:

print(num,"is not an Armstrong number")

Output 1

Enter a number: 663

663 is not an Armstrong number

Output 2

Enter a number: 407

407 is an Armstrong number

Here, we ask the user for a number and check if it is an Armstrong number.

We need to calculate the sum of cube of each digit. So, we initialize the sum to 0 and obtain each digit number by using the [modulus operator %](http://www.programiz.com/python-programming/operators#arithmetic_operators). Remainder of a number when it is divide by 10 is the last digit of that number. We take the cubes using exponent operator.

Finally, we compare the sum with the original number and conclude that it is Armstrong number if they are equal.

## Source Code: Check Armstrong number of n digits

num = 1634

# Changed num variable to string,

# and calculated the length (number of digits)

order = len(str(num))

# initialize sum

sum = 0

# find the sum of the cube of each digit

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

# display the result

if num == sum:

print(num,"is an Armstrong number")

else:

print(num,"is not an Armstrong number")

You can change the value of num in the source code and run again to test it.