


Basic programs

Python Programming Examples - GeeksforGeeks

The following Python section contains a wide collection of Python programming examples. The examples are categorized based on the topics including List, strings, dictionary, tuple, sets, and many more.

 <https://www.geeksforgeeks.org/python-programming-examples/>



Get a list as input from user

```
# creating an empty list
lst = []

# number of elements as input
n = int(input("Enter number of elements : "))

# iterating till the range
for i in range(0, n):
    ele = int(input())

    lst.append(ele) # adding the element

print(lst)
```

Sum of list

```
# Python program to find sum of elements in list
total = 0

# creating a list
list1 = [11, 5, 17, 18, 23]

# Iterate each element in list
# and add them in variable total
for ele in range(0, len(list1)):
    total = total + list1[ele]
```

```
# printing total value
print("Sum of all elements in given list: ", total)
```

Find duplicates and duplicate count for the given list

```
a = [1,2,3,2,1,5,6,5,5,5]

import collections
print([item for item, count in collections.Counter(a).items() if count > 1])

## [1, 2, 5]

def count_duplicates(seq):

    '''takes as argument a sequence and
    returns the number of duplicate elements'''

    return len(seq) - len(set(seq))

res = count_duplicates([-1,2,4,2,0,4,4])
print(res) # -> 3
```

```
#lines: Input, assignment

name = input('What is your name?\n')
print ('Hi, %s.' % name)

# For loop, built-in enumerate function, new style formatting

friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    print ("iteration {iteration} is {name}".format(iteration=i, name=name))

# Functions

def greet(name):
    print ('Hello', name)

greet('Jack')
greet('Jill')
greet('Bob')
```

```
# Dictionaries, generator expressions

prices = {'apple': 0.40, 'banana': 0.50}
my_purchase = {
    'apple': 1,
    'banana': 6}
grocery_bill = sum(prices[fruit] * my_purchase[fruit]
                   for fruit in my_purchase)
print ('I owe the grocer $%.2f' % grocery_bill)
```

Time, conditionals, from..import, for..else

```
from time import localtime

activities = {8: 'Sleeping',
              9: 'Commuting',
              17: 'Working',
              18: 'Commuting',
              20: 'Eating',
              22: 'Resting' }

time_now = localtime()
hour = time_now.tm_hour

for activity_time in sorted(activities.keys()):
    if hour < activity_time:
        print (activities[activity_time])
        break
else:
    print ('Unknown, AFK or sleeping!')
```

Python3 program to swap first and last element of a list

```
# Swap function
def swapList(newList):

    newList[0], newList[-1] = newList[-1], newList[0]

    return newList

# Driver code
newList = [12, 35, 9, 56, 24]
print(swapList(newList))
```

```

# Python3 program to swap elements
# at given positions

def swapPositions(list, pos1, pos2):
    list[pos1],list[pos2] = list[pos2],list[pos1]
    return list

# Driver Code
List = [23, 65, 19, 90]
pos1, pos2 = 1, 3
print(swapPositions(List, pos1-1, pos2-1))

# Reversing a list using slicing technique
def Reverse(lst):
    new_lst = lst[::-1]
    return new_lst

lst = [10, 11, 12, 13, 14, 15]
print(Reverse(lst))

# Python program to find largest
# number in a list

# list of numbers
list1 = [10, 20, 4, 45, 99]

# sorting the list
list1.sort()

# printing the last element
print("Largest element is:", list1[-1])

```

```

>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> spam[0:4]
['cat', 'bat', 'rat', 'elephant']
>>> spam[1:3]
['bat', 'rat']
>>> spam[0:-1]
['cat', 'bat', 'rat']

```

Remove Duplicates

```

def remove_duplicate(list1):
    list11 = []
    for item in list1:

```

```

        if item not in list11:
            list11.append(item)
        return list11

print(remove_duplicate([1,2,3,2,3,1]))

```

```

# Function to reverse words of string

def rev_sentence(sentence):

    # first split the string into words
    words = sentence.split(' ')

    # then reverse the split string list and join using space
    reverse_sentence = ' '.join(reversed(words))

    # finally return the joined string
    return reverse_sentence

if __name__ == "__main__":
    input = 'geeks quiz practice code'
    print (rev_sentence(input))

```

```

# function which return reverse of a string

def isPalindrome(s):
    return s == s[::-1]

# Driver code
s = "malayalam"
ans = isPalindrome(s)

if ans:
    print("Yes")
else:
    print("No")

```

```

# Python 3 program for recursive binary search.
# Modifications needed for the older Python 2 are found in comments.

# Returns index of x in arr if present, else -1
def binary_search(arr, low, high, x):

    # Check base case
    if high >= low:

```

```

    mid = (high + low) // 2

    # If element is present at the middle itself
    if arr[mid] == x:
        return mid

    # If element is smaller than mid, then it can only
    # be present in left subarray
    elif arr[mid] > x:
        return binary_search(arr, low, mid - 1, x)

    # Else the element can only be present in right subarray
    else:
        return binary_search(arr, mid + 1, high, x)

else:
    # Element is not present in the array
    return -1

# Test array
arr = [ 2, 3, 4, 10, 40 ]
x = 10

# Function call
result = binary_search(arr, 0, len(arr)-1, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")

```

```

# Python3 program to find Minimum
# number of jumps to reach end

# Returns minimum number of jumps
# to reach arr[h] from arr[l]
def minJumps(arr, l, h):

    # Base case: when source and
    # destination are same
    if (h == l):
        return 0

    # when nothing is reachable
    # from the given source
    if (arr[l] == 0):
        return float('inf')

    # Traverse through all the points
    # reachable from arr[l]. Recursively
    # get the minimum number of jumps
    # needed to reach arr[h] from

```

```

# these reachable points.
min = float('inf')
for i in range(l + 1, h + 1):
    if (i < l + arr[l] + 1):
        jumps = minJumps(arr, i, h)
        if (jumps != float('inf') and
            jumps + 1 < min):
            min = jumps + 1

return min


# Driver program to test above function
arr = [1, 3, 6, 3, 2, 3, 6, 8, 9, 5]
n = len(arr)
print('Minimum number of jumps to reach',
      'end is', minJumps(arr, 0, n-1))

```

FizzBuzz

Fizz Buzz in Python

"Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both

 <https://k3no.medium.com/fizz-buzz-in-python-2edea331d715>



```

for i in range(1, 101):
    if i % 3 == 0 and i % 5 == 0:
        print ('FizzBuzz')
    elif i % 3 == 0:
        print ('Fizz')
    elif i % 5 == 0:
        print ('Buzz')
    else:
        print (i)

for i in range(1,101):
    fizz = 'Fizz' if i%3==0 else ''
    buzz = 'Buzz' if i%5==0 else ''
    print(f'{fizz}{buzz}' or i)

print(*map(lambda i: 'Fizz'*(not i%3)+'Buzz'*(not i%5) or i, range(1,101)),sep='\n')s

```

Least Common Multiple(LCM)

LCM (Least Common Multiple) of two numbers is the smallest number which can be divided by both numbers.

For example, LCM of 15 and 20 is $5 \times 3 \times 5 \times 4 = 5 \times 3 \times 4 = 60$,

and LCM of 5 and 7 is 35.

LCM and HCF

LCM using Repeated Division
Find the LCM of 24 and 36

2	24	36
2	12	18
3	6	9
	2	3

LCM: $2 \times 2 \times 3 \times 2 \times 3 = 72$

HCF using Repeated Division
Find the HCF of 24 and 36

2	24	36
2	12	18
3	6	9
	2	3

HCF: $2 \times 2 \times 3 = 12$

```
# Python program to find LCM of two numbers

# Recursive function to return gcd of a and b
def gcd(a,b):
    if a == 0:
        return b
    return gcd(b % a, a)

# Function to return LCM of two numbers
def lcm(a,b):
    return (a / gcd(a,b)) * b

# Driver program to test above function
a = 15
b = 20
print('LCM of', a, 'and', b, 'is', lcm(a, b))
```



```

# Python code to demonstrate the working of gcd()
# importing "math" for mathematical operations
import math

# prints 12
print("The gcd of 60 and 48 is : ", end="")
print(math.lcm(60, 48))

import numpy

>>> np.lcm(25,45)
225
>>> np.gcd(25,45)
5

```

Convert dictionary with numpy array to json and back

```

numpy arrays cannot be converted into JSON directly; instead, use list.

# Test data
d = {
    'chicken': np.random.randn(5),
    'banana': np.random.randn(5),
    'carrots': np.random.randn(5)
}

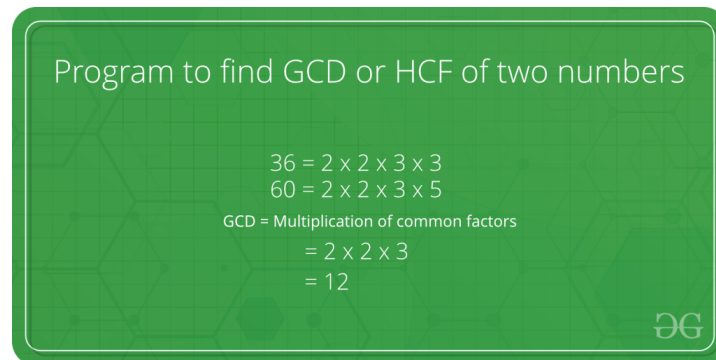
# To json
j = json.dumps({k: v.tolist() for k, v in d.items()})

# Back to dict
a = {k: np.array(v) for k, v in json.loads(j).items()}

print (a)
print (d)

```

Greatest Common Divisor (GCD)



```
# Python code to demonstrate naive
# method to compute gcd ( recursion )

def hcf(a, b):
    if(b == 0):
        return a
    else:
        return hcf(b, a % b)

a = 60
b = 48

# prints 12
print("The gcd of 60 and 48 is : ", end="")
print(hcf(60, 48))

O/P:
12
```

Factorial

Factorial of a non-negative integer is the multiplication of all integers smaller than or equal to n. For example factorial of 6 is $6 \times 5 \times 4 \times 3 \times 2 \times 1$ which is 720.

```
# Recursive
def factorial(n):
    # single line to find factorial
    return 1 if (n==1 or n==0) else n * factorial(n - 1);
num = 5;
```

```

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

def FirstFactorial(num):
    if num == 1:
        return 1
    else:
        return num * FirstFactorial(num-1)
    # code goes here
    # return num

# keep this function call here
print(FirstFactorial(input()))

```

```

#Iterative
def factorial(n):

    # single line to find factorial
    return 1 if (n==1 or n==0) else n * factorial(n - 1);

# Driver Code
num = 5;
print("Factorial of",num,"is",
factorial(num))

```

```

# One line Solution (Using Ternary operator):
def factorial(n):

    # single line to find factorial
    return 1 if (n==1 or n==0) else n * factorial(n - 1)

# Driver Code
num = 5
print ("Factorial of",num,"is",
      factorial(num))

```

Armstrong Number

Given a number x , determine whether the given number is Armstrong number or not. A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

$abcd... = \text{pow}(a,n) + \text{pow}(b,n) + \text{pow}(c,n) + \text{pow}(d,n) +$

Example:

Input : 153

Output : Yes

153 is an Armstrong number.

$1^3 + 5^3 + 3^3 = 153$

```
# Function to calculate x raised to the power y

def power(x, y):
    if y==0:
        return 1
    if y%2==0:
        return power(x, y/2)*power(x, y/2)
    return x*power(x, y/2)*power(x, y/2)

# Function to calculate the order of the number

def order(x):

    # variable to store of the number
    n = 0
    while (x!=0):
        n = n+1
        x = x/10
    return n

# Function to check whether the given number is

# Armstrong number or not

def isArmstrong (x):
    n = order(x)
    temp = x
    sum1 = 0
    while (temp!=0):
        r = temp%10
        sum1 = sum1 + power(r, n)
        temp = temp/10

        # If condition satisfies
        return (sum1 == x)

# Driver Program

x = 153
print(isArmstrong(x))
x = 1253
print(isArmstrong(x))
```

Fibonacci Series

Python Program for Fibonacci numbers

The Fibonacci numbers are the numbers in the following integer sequence.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,

In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the recurrence relation

$$F_n = F_{n-1} + F_{n-2}$$

with seed values

$F_0 = 0$ and $F_1 = 1$.

```
Method 1 ( Use recursion ) :
# Function for nth Fibonacci number

def Fibonacci(n):
    if n<0:
        print("Incorrect input")
    # First Fibonacci number is 0
    elif n==1:
        return 0
    # Second Fibonacci number is 1
    elif n==2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

# Driver Program

print(Fibonacci(9))
```

```
#Method 2 ( Space Optimized):

# Function for nth fibonacci number - Space Optimisatation
# Taking 1st two fibonacci numbers as 0 and 1
```

```

def fibonacci(n):
    a = 0
    b = 1
    if n < 0:
        print("Incorrect input")
    elif n == 0:
        return a
    elif n == 1:
        return b
    else:
        for i in range(2,n):
            c = a + b
            a = b
            b = c
        return b

# Driver Program

print(fibonacci(9))

```

To find the second largest number in a list

Examples:

Input : list1 = [10, 20, 4]

Output : 10

Input : list2 = [70, 11, 20, 4, 100]

Output: 70

```

#Method 1: Sorting is an easier but less optimal method.
#Given below is an O(n) algorithm to do the same.
# list of numbers - length of the list should be at least 2
list1 = [10, 20, 4, 45, 99]

max=max(list1[0],list1[1])
secondmax=min(list1[0],list1[1])

for i in range(2,len(list1)):
    if list1[i]>max:
        secondmax=max
        max=list1[i]
    else:
        if list1[i]>secondmax:
            secondmax=list1[i]

print("Second highest number is : ",str(secondmax))

```

```
#Method 2: Sort the list in ascending order and print the second last  
# element in the list.
```

```
# Python program to find largest  
# number in a list
```

```
# list of numbers  
list1 = [10, 20, 4, 45, 99]
```

```
# sorting the list  
list1.sort()
```

```
# printing the second last element  
print("Second largest element is:", list1[-2])  
Output:
```

```
Largest element is: 45
```

Python program to check if a string is palindrome or not

Given a string, write a python function to check if it is palindrome or not. A string is said to be palindrome if the reverse of the string is the same as string. For example, "radar" is a palindrome, but "radix" is not a palindrome.

Examples:

Input : malayalam

Output : Yes

Input : geeks

Output : No

```
#1) Find reverse of a string  
#2) Check if reverse and original are the same or not.
```

```

# function which return reverse of a string
def reverse(s):
    return s[::-1]

def isPalindrome(s):
    # Calling reverse function
    rev = reverse(s)

    # Checking if both string are equal or not
    if (s == rev):
        return True
    return False

# Driver code
s = "malayalam"
ans = isPalindrome(s)

if ans == 1:
    print("Yes")
else:
    print("No")
Output :

```

Longest Word

Have the function LongestWord(**sen**) take the **sen** parameter being passed and return the largest word in the string. If there are two or more words that are the same length, return the first word from the string with that length. Ignore punctuation and assume **sen** will not be empty.

```

import re
pattern = re.compile(r'\W+')
def LongestWord(sen):
    x = pattern.split(sen)
    return max(x, key=len)

# keep this function call here
print(LongestWord(input()))

```

Reversed LinkedList


```

Python program to reverse a linked list
# Time Complexity : O(n)
# Space Complexity : O(1)

# Node class
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # Function to reverse the linked list
    def reverse(self):
        prev = None
        current = self.head
        while(current is not None):
            next = current.next
            current.next = prev
            prev = current
            current = next
        self.head = prev

    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Utility function to print the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print temp.data,
            temp = temp.next

# Driver program to test above functions
llist = LinkedList()
llist.push(20)
llist.push(4)
llist.push(15)
llist.push(85)

print "Given Linked List"
llist.printList()
llist.reverse()
print "\nReversed Linked List"
llist.printList()

```

Given a set of strings, write a Python program to determine common prefix from a set of strings.

```
# Initialising string
ini_strlist = ['akshat', 'akash', 'akshay', 'akshita']

# Finding common prefix using Naive Approach
res = ''
prefix = ini_strlist[0]

for string in ini_strlist[1:]:
    while string[:len(prefix)] != prefix and prefix:
        prefix = prefix[:len(prefix)-1]
    if not prefix:
        break
res = prefix

# Printing result
print("Resultant prefix", str(res))
```

Maximum Product SubArray

Maximum Product Subarray - GeeksforGeeks

Given an array that contains both positive and negative integers, find the product of the maximum product subarray. Expected Time complexity is $O(n)$ and only $O(1)$ extra space can be used.

 <https://www.geeksforgeeks.org/maximum-product-subarray/>



```
# Python3 program to find Maximum Product Subarray

# Returns the product of max product subarray.
def maxSubarrayProduct(arr, n):

    # Initializing result
    result = arr[0]

    for i in range(n):

        mul = arr[i]

        # traversing in current subarray
```

```

        for j in range(i + 1, n):

            # updating result every time
            # to keep an eye over the maximum product
            result = max(result, mul)
            mul *= arr[j]

        # updating the result for (n-1)th index.
        result = max(result, mul)

    return result

# Driver code
arr = [ 1, -2, -3, 0, 7, -8, -2 ]
n = len(arr)
print("Maximum Sub array product is" , maxSubarrayProduct(arr, n))

```

Stock Maximize solution

Hackerrank - Stock Maximize Solution

Your algorithms have become so good at predicting the market that you now know what the share price of Wooden Orange Toothpicks Inc. (WOT) will be for the next number of days. Each day, you can

🔗 <https://www.thepoorcoder.com/hackerrank-stock-maximize-solution/>



```

def stockmax(prices):
    m = prices.pop()
    maxsum = 0
    arrsum = 0
    for p in reversed(prices):
        m = max(m, p)
        maxsum+=m
        arrsum+=p
    return maxsum-arrsum

for _ in range(int(input())):
    n = int(input())
    prices = list(map(int,input().split()))
    print(stockmax(prices))

```

Convert tuple into dictionary

```
# Python code to convert into dictionary


def Convert(tup, di):
    for a, b in tup:
        di.setdefault(a, []).append(b)
    return di

# Driver Code
tups = [("akash", 10), ("gaurav", 12), ("anand", 14),
        ("suraj", 20), ("akhil", 25), ("ashish", 30)]
dictionary = {}
print (Convert(tups, dictionary))
```

Sort the values of one list using the second list

What is the purpose of the single underscore "_" variable in Python?

has 3 main conventional uses in Python: To hold the result of the last executed expression(/statement) in an interactive interpreter session (see docs).

 <https://stackoverflow.com/questions/5893163/what-is-the-purpose-of-the-single-underscore-variable-in-python>



```
# sort the values of one list using the second list.
list1 = ["a", "b", "c", "d", "e", "f", "g", "h", "i"]
list2 = [ 0,  1,  1,  0,  1,  2,  2,  0,  1]
```

```
l1 = zip(list2, list1)
```

```
z = [x for _, x in sorted(l1)]
```

```
print(z)
```

o/p:

```
['a', 'd', 'h', 'b', 'c', 'e', 'i', 'f', 'g']
```

```
# Python code to demonstrate the working of
# zip()
```

```
# initializing lists
```

```

name = [ "Manjeet", "Nikhil", "Shambhavi", "Astha" ]
roll_no = [ 4, 1, 3, 2 ]
marks = [ 40, 50, 60, 70 ]

# using zip() to map values
mapped = zip(name, roll_no, marks)

# converting values to print as set
mapped = set(mapped)

# printing resultant values
print ("The zipped result is : ",end="")
print (mapped)

o/p:

The zipped result is : {('Shambhavi', 3, 60), ('Astha', 2, 70),
('Manjeet', 4, 40), ('Nikhil', 1, 50)}

```

Find the second last element

```

list2 = [70, 11, 20, 4, 100]

list2.sort()

print(list2[-2])

# Python program to find second largest
# number in a list

# list of numbers
list1 = [10, 20, 4, 45, 99]

# new_list is a set of list1
new_list = set(list1)

# removing the largest element from temp list
new_list.remove(max(new_list))

# elements in original list are not changed
# print(list1)

print(max(new_list))

```

Command Line Argument

```
# Python program to demonstrate
# command line arguments

import sys

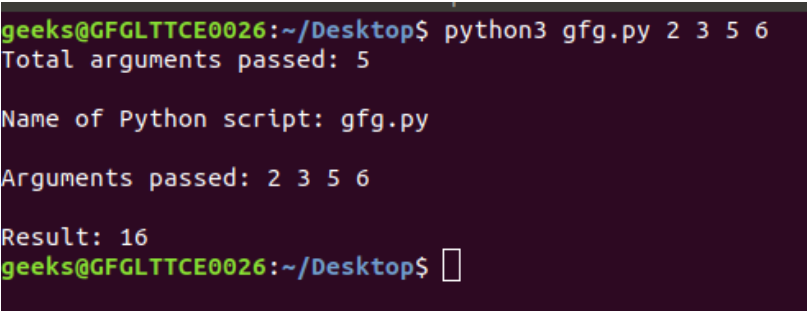
# total arguments
n = len(sys.argv)
print("Total arguments passed:", n)

# Arguments passed
print("\nName of Python script:", sys.argv[0])

print("\nArguments passed:", end = " ")
for i in range(1, n):
    print(sys.argv[i], end = " ")

# Addition of numbers
Sum = 0
# Using argparse module
for i in range(1, n):
    Sum += int(sys.argv[i])

print("\n\nResult:", Sum)
```



```
geeks@GFGLTTCE0026:~/Desktop$ python3 gfg.py 2 3 5 6
Total arguments passed: 5

Name of Python script: gfg.py

Arguments passed: 2 3 5 6

Result: 16
geeks@GFGLTTCE0026:~/Desktop$
```

Rock, Paper, Scissors

```

import random, sys
print('ROCK, PAPER, SCISSORS')
# These variables keep track of the number of wins, losses, and ties.
wins = 0
losses = 0
ties = 0
while True: # The main game loop.
    print('%s Wins, %s Losses, %s Ties' % (wins, losses, ties))
    while True: # The player input loop.
        print('Enter your move: (r)ock (p)aper (s)ciissors or (q)uit')
        playerMove = input()
        if playerMove == 'q':
            sys.exit() # Quit the program.
        if playerMove == 'r' or playerMove == 'p' or playerMove == 's':
            break # Break out of the player input loop.
        print('Type one of r, p, s, or q.')

    # Display what the player chose:
    if playerMove == 'r':
        print('ROCK versus...')
    elif playerMove == 'p':
        print('PAPER versus...')
    elif playerMove == 's':
        print('SCISSORS versus...')

    # Display what the computer chose:
    randomNumber = random.randint(1, 3)
    if randomNumber == 1:
        computerMove = 'r'
        print('ROCK')
    elif randomNumber == 2:
        computerMove = 'p'
        print('PAPER')
    elif randomNumber == 3:
        computerMove = 's'
        print('SCISSORS')

    # Display and record the win/loss/tie:
    if playerMove == computerMove:
        print('It is a tie!')
        ties = ties + 1
    elif playerMove == 'r' and computerMove == 's':
        print('You win!')
        wins = wins + 1
    elif playerMove == 'p' and computerMove == 'r':
        print('You win!')
        wins = wins + 1
    elif playerMove == 's' and computerMove == 'p':
        print('You win!')
        wins = wins + 1
    elif playerMove == 'r' and computerMove == 'p':
        print('You lose!')

```

```
losses = losses + 1
elif playerMove == 'p' and computerMove == 's':
    print('You lose!')
    losses = losses + 1
elif playerMove == 's' and computerMove == 'r':
    print('You lose!')
    losses = losses + 1
```