Python Program for Find minimum sum of factors of number

```
def findMinSum(num):
    sum = 0

    # Find factors of number
    # and add to the sum
    i = 2
    while(i * i <= num):
        while(num % i == 0):
            sum += i
            num /= i
        i += 1
    sum += num

    # Return sum of numbers
    # having minimum product
    return sum

# Driver Code
num = 15
print(int(findMinSum(num)))
```

```
8
```

Write a python program to check a given number is perfect or not.

Note: A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. Lets understand it with example

6 is a positive number and its divisor is 1,2,3 and 6 itself.

But we should not include 6 as by the definition of perfect number.

Lets add its divisor excluding itself

1+2+3 = 6 which is equal to number itself.

It means 6 is a Perfect Number.

```
n = int(input("Enter any number: "))
sum1 = 0
for i in range(1, n):
    if(n % i == 0):
        sum1 = sum1 + i
if (sum1 == n):
    print("The number is a Perfect number!")
else:
    print("The number is not a Perfect number!")
```

```
Enter any number: 6
The number is a Perfect number!
```

Write a Python program to check whether a String is Palindrome or not?

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


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

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

```
Yes
```

Remove char. from string Easy Write a python program to remove a given character from string.

For Example:

Input String: Quescol

Input Character : e

Output: Quscol (After removing 'e')

```
def remove_char(s1,s2):
    print(s1.replace(s2, ''))

s1 = input("please give a String : ")
s2 = input("please give a Character to remove : ")
remove_char(s1,s2)
```

```
please give a String : almabetter
please give a Character to remove : t
almabeer
```

Swap two numbers Easy Write a Python program to swap two number without using third variable.

```
x = 5
y = 7

print ("Before swapping: ")
print("Value of x : ", x, " and y : ", y)

# code to swap 'x' and 'y'
x, y = y, x

print ("After swapping: ")
print("Value of x : ", x, " and y : ", y)
```

```
Before swapping:
Value of x :  5  and y :  7
After swapping:
Value of x :  7  and y :  5
```

Find last position of substring

Write a python program to find the last position of a given substring.

```
# initializing string
test_string = "GfG is best for CS and also best for Learning"

# initializing target word
tar_word = "best"

# printing original string
print("The original string : " + str(test_string))

# using rindex()
# Find last occurrence of substring
res = test_string.rindex(tar_word)

# print result
print("Index of last occurrence of substring is : " + str(res))
```

```
The original string : GfG is best for CS and also best for Learning
Index of last occurrence of substring is : 28
```

Change position Easy Write a Python program to change the position of every nth value with the (n+1)th in a list.

```
from itertools import zip_longest, chain, tee
def replace2copy(lst):
    lst1, lst2 = tee(iter(lst), 2)
    return list(chain.from_iterable(zip_longest(lst[1::2], lst[::2])))
n = [0,1,2,3,4,5]
print(replace2copy(n))
```

```
[1, 0, 3, 2, 5, 4]
```

Common ones out

Write a Python program to find common items from two lists.

```
color1 = "Red", "Green", "Orange", "White"
color2 = "Black", "Green", "White", "Pink"
print(set(color1) & set(color2))
```

```
{'Green', 'White'}
```

Difference between two lists

Write a Python program to compute the difference between two lists.

```
from collections import Counter
color1 = ["red", "orange", "green", "blue", "white"]
color2 = ["black", "yellow", "green", "blue"]
counter1 = Counter(color1)
counter2 = Counter(color2)
print("Color1-Color2: ",list(counter1 - counter2))
print("Color2-Color1: ",list(counter2 - counter1))
```

```
Color1-Color2:  ['red', 'orange', 'white']
Color2-Color1:  ['black', 'yellow']
```

Create lambda function

Write a Python program to create a lambda function that adds 15 to a given number passed in as an argument, also create a lambda function that multiplies argument x with argument y and print the result.

```
r = lambda a : a + 15
print(r(10))
r = lambda x, y : x * y
print(r(12, 4))
```

```
25
48
```

Find Tuple with smallest index

Write a Python program to find a tuple, the smallest second index value from a list of tuples.

```
x = [(4, 1), (1, 2), (6, 0)]
print(min(x, key=lambda n: (n[1], -n[0])))
```

```
(6, 0)
```

Find items starts with char.

Write a Python program to find the items starts with specific character from a given list.

```
def test(lst, char):
    result = [i for i in lst if i.startswith(char)]
    return result
text = ["abcd", "abc", "bcd", "bkie", "cder", "cdsw", "sdfsd", "dagfa", "acjd"]
print("\nOriginal list:")
print(text)
char = "a"
print("\nItems start with",char,"from the said list:")
print(test(text, char))
```

```
Original list:
```

```
['abcd', 'abc', 'bcd', 'bkie', 'cder', 'cdsw', 'sdfsd', 'dagfa', 'acjd']

Items start with a from the said list:
['abcd', 'abc', 'acjd']
```

Using lambda find items begin with char.

Write a Python program to find whether a given string starts with a given character using Lambda.

```
starts_with = lambda x: True if x.startswith('P') else False
print(starts_with('Python'))
starts_with = lambda x: True if x.startswith('P') else False
print(starts_with('Java'))
```

```
True
False
```

Find next prime number

Write a Python Program to Find Next Prime Number.

```
def nextprime(n):
    p=n+1
    for i in range(2,p):
        if(p%i==0):
            p=p+1
    else:
        print(p,end=" ")
n=int(input())
nextprime(n)
```

```
9
11
```

Calculate Euclidean distance

Write a Python program to calculate Euclidean Distance.

```
import math
print("Enter the first point A")
x1, y1 = map(int, input().split())
print("Enter the second point B")
x2, y2 = map(int, input().split())
dist = math.sqrt((x2-x1)**2 + (y2-y1)**2)
print("The Euclidean Distance is " + str(dist))
```

```
Enter the first point A
5 6
Enter the second point B
6 7
The Euclidean Distance is 1.4142135623730951
```

Filter sublist using lambda

Write a Python program to filter a list of integers using Lambda.

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("Original list of integers:")
print(nums)
print("\nEven numbers from the said list:")
even_nums = list(filter(lambda x: x%2 == 0, nums))
print(even_nums)
print("\nOdd numbers from the said list:")
odd_nums = list(filter(lambda x: x%2 != 0, nums))
print(odd_nums)
```

```
Original list of integers:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Even numbers from the said list:
[2, 4, 6, 8, 10]

Odd numbers from the said list:
[1, 3, 5, 7, 9]
```

Armstrong number Easy Write a program in Python to check whether an integer is Armstrong number or not.

What is Armstrong Number?

It is a number which is equal to the sum of cube of its digits. For eg: 153, 370 etc.

Lets see it practically by calculating it,

Suppose I am taking 153 for an example

First calculate the cube of its each digits

1^3 = (1 * 1 * 1) = 1

5^3 = (5 * 5 * 5) = 125

3^3= (3 * 3 * 3) = 27

Now add the cube

1+125+27 = 153

It means 153 is an Armstrong Number.

```
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")
```

```
Enter a number: 153
153 is an Armstrong number
```

Largest negative and smallest positive Easy Write a Python program to find the largest negative and smallest positive numbers (or 0 if none).

```
def test(nums):
    pos = [n for n in nums if n > 0]
    neg = [n for n in nums if n < 0]
    return [max(neg) if neg else 0, min(pos) if pos else 0]
nums=[-12, -6, 300, -40, 2, 2, 3, 57, -50, -22, 12, 40, 9, 11, 18]
print("List of numbers:",nums)
print("Largest negative and smallest positive numbers (or 0 if none) of the said list:")
print(test(nums))
```

```
List of numbers: [-12, -6, 300, -40, 2, 2, 3, 57, -50, -22, 12, 40, 9, 11, 18]
Largest negative and smallest positive numbers (or 0 if none) of the said list:
[-6, 2]
```

Harmonic sum

Write a Python program to calculate the harmonic sum of n-1.

Note: The harmonic sum is the sum of reciprocals of the positive integers.

```
def harmonic_sum(n):
  if n < 2:
    return 1
  else:
    return 1 / n + (harmonic_sum(n - 1))

print(harmonic_sum(7))
print(harmonic_sum(4))
```

```
2.5928571428571425
2.083333333333333
```

Largest prime factor Easy Write a Python program to find the largest prime factor of a given number.

The prime factors of 330 are 2, 3, 5 and 11.

Therefore 11 is the largest prime factor of 330.

Sample Example: 330 = 2 × 3 × 5 × 11

```
def Largest_Prime_Factor(n):
    return next(n // i for i in range(1, n) if n % i == 0 and is_prime(n // i))
def is_prime(m):
    return all(m % i for i in range(2, m - 1))
print(Largest_Prime_Factor(200))
print(Largest_Prime_Factor(330))
print(Largest_Prime_Factor(243423423330))
```

```
5
11
1114117
```

Add digits

Write a Python program to add the digits of a positive integer repeatedly until the result has a single digit.

```
def add_digits(num):
        return (num - 1) % 9 + 1 if num > 0 else 0

print(add_digits(48))
print(add_digits(59))
```

```
3
5
```

Find GCD

Write a Python program to find the greatest common divisor (GCD) of two integers,

```
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))
```

```
The gcd of 60 and 48 is : 12
```

Biggest even number Easy Write a Python program to find the biggest even number between two numbers inclusive.

```
def test(m, n):
    if m > n or (m == n and m % 2 == 1):
        return -1
    return n if n % 2 == 0 else n - 1
```

```
m = 12
n = 51
print("\nBiggest even number between",m,"and",n)
print(test(m, n))
```

```
Biggest even number between 12 and 51
50
```

Find x

Write a Python program to find an integer exponent x such that a^x = n.

Input:

n = 81

a = 3

Output: 4

```
def test(n,a):
    m = 1
    x = 0
    while m != n:
        x += 1
        m *= a
    return x

a = 3
n = 81
print("a = ",a,": n = ",n)
print("Find an integer exponent x such that a^x = n:")
print(test(n,a))
```

```
a =  3 : n =  81
Find an integer exponent x such that a^x = n:
4
```

Words with prime length

Write a Python program to find the string consisting of all the words whose lengths are prime numbers.

```
def test(strs):
    return " ".join(strs for strs in strs.split() if is_prime(len(strs)))
def is_prime(n):
    return n > 1 and all(n % j for j in range(2, int(n ** 0.5) + 1))
strs = "The quick brown fox jumps over the lazy dog."
print("Original list of numbers:")
print(strs)
print("Words whose lengths are prime numbers in the said string:")
print(test(strs))
```

```
Original list of numbers:
The quick brown fox jumps over the lazy dog.
```

```
Words whose lengths are prime numbers in the said string:
The quick brown fox jumps the
```

Make sub-list of only odd numbers

Write a Python program to find the sublist of numbers from a given list of numbers with only odd digits in increasing order.

```
def test(nums):
    return sorted(n for n in nums if all(int(c) % 2 for c in str(abs(n))))

nums = [1, 3, 79, 10, 4, 2, 39]
print("Original list of numbers:")
print(nums)
print("Sublist of numbers of the said list with only odd digits in increasing order:")
print(test(nums))
```

```
Original list of numbers:
[1, 3, 79, 10, 4, 2, 39]
Sublist of numbers of the said list with only odd digits in increasing order:
[1, 3, 39, 79]
```

Factorial by iterative method

Write a Python program to calculate the factorial using iterative approach.

```
num=5
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)
```

```
The factorial of 5 is 120
```

Fibonacci by Iterative method

Write a Fibonacci series program in python using iterative method.

```
def fibonacci(n):
    assert n>0
    secondLast = 0
    Last = 1
    if n == 1:
        print(secondLast)
    elif n == 2:
        print(Last)
```

```
        else:
            for i in range(3, n+1):
                result = secondLast + Last
                secondLast = Last
                Last = result
            print(result)


    fibonacci(5)
    fibonacci(10)
```

```
    3
    34
```

```
    first,second=0,1
    n = int(input("please give a number for fibonacci series : "))
    print("fibonacci series are : ")
    for i in range(0,n):
        if i<=1:
            result=i
        else:
          result = first + second;
          first = second;
          second = result;
        print(result,end=',')
```

```
    please give a number for fibonacci series : 10
    fibonacci series are :
    0,1,1,2,3,5,8,13,21,34,
```

Palindrome by iterative method

Write a Python program for palindrome using an iterative method.

```
    n = int(input("please give a number : "))
    reverse,temp = 0,n
    while temp!=0:
        reverse = reverse*10 + temp%10;
        temp=temp//10;
    if reverse==n:
        print("number is palindrom")
    else:
        print("number is not palindrom")
```

```
    please give a number : 5885
    number is palindrom
```

Remove duplicates from string

Write a Python program to remove duplicate characters of a given string.

```
from collections import OrderedDict
def remove_duplicate(str1):
  return "".join(OrderedDict.fromkeys(str1))

print(remove_duplicate("python exercises practice solution"))
print(remove_duplicate("w3resource"))
```

```
python exrcisalu
w3resouc
```

Count freq of chars

Write a Python program to count the number of characters (character frequency) in a string.

```
def char_frequency(str1):
    dict = {}
    for n in str1:
        keys = dict.keys()
        if n in keys:
            dict[n] += 1
        else:
            dict[n] = 1
    return dict
print(char_frequency('google.com'))
```

```
{'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}
```

Swap first two char from given string

Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.

```
def chars_mix_up(a, b):
  new_a = b[:2] + a[2:]
  new_b = a[:2] + b[2:]

  return new_a + ' ' + new_b
print(chars_mix_up('abc', 'xyz'))
```

```
xyc abz
```

Change to $

Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.

```
def change_char(str1):
  char = str1[0]
  length = len(str1)
  str1 = str1.replace(char, '$')
  str1 = char + str1[1:]
```

```
  return str1

print(change_char('restart'))
```

```
resta$t
```

Count words in sentence

Write a Python program to count the occurrences of each word in a given sentence.

```
def word_count(str):
    counts = dict()
    words = str.split()

    for word in words:
        if word in counts:
            counts[word] += 1
        else:
            counts[word] = 1

    return counts

print( word_count('the quick brown fox jumps over the lazy dog.'))
```

```
{'the': 2, 'quick': 1, 'brown': 1, 'fox': 1, 'jumps': 1, 'over': 1, 'lazy': 1, 'dog.': 1}
```

Print unique words

Write a Python program that accepts a comma separated sequence of words as input and prints the unique words in sorted form (alphanumerically).

```
items = input("Input comma separated sequence of words")
words = [word for word in items.split(",")]
print(",".join(sorted(list(set(words)))))
```

```
Input comma separated sequence of words red, black, pink, green, black, green
 black, green, pink, red
```

Insert string

Write a Python function to insert a string in the middle of a string.

```
def insert_sting_middle(str, word):
    return str[:2] + word + str[2:]

print(insert_sting_middle('[[]]', 'Python'))
print(insert_sting_middle('{{}}', 'PHP'))
print(insert_sting_middle('<<>>', 'HTML'))
```

```
[[Python]]
{{PHP}}
```

```
<<HTML>>
```

Reverse a string

Write a recursive Python function to reverse a string if it's length is a multiple of 4.

```
def reverse_string(str1):
    if len(str1) % 4 == 0:
       return ''.join(reversed(str1))
    return str1

print(reverse_string('abcd'))
print(reverse_string('python'))
```

```
dcba
python
```

Print first non-repeating char

Write a Python program to find the first non-repeating character in given string.

```
def first_non_repeating_character(str1):
  char_order = []
  ctr = {}
  for c in str1:
    if c in ctr:
      ctr[c] += 1
    else:
      ctr[c] = 1
      char_order.append(c)
  for c in char_order:
    if ctr[c] == 1:
      return c
  return None

print(first_non_repeating_character('abcdef'))
print(first_non_repeating_character('abcabcdef'))
print(first_non_repeating_character('aabbcc'))
```

```
a
d
None
```

Print second most repeated word

Write a Python program to find the second most repeated word in a given string.

```
def word_count(str):
    counts = dict()
    words = str.split()
```

```
    for word in words:
        if word in counts:
            counts[word] += 1
        else:
            counts[word] = 1

    counts_x = sorted(counts.items(), key=lambda kv: kv[1])
    #print(counts_x)
    return counts_x[-2]

print(word_count("Both of these issues are fixed by postponing the evaluation of annotations. Instead of compiling code which executes expressions in annotations at
```

```
('of', 4)
```

Sort list of tuples

Write a Python program to get a list, sorted in increasing order by the last element in each tuple from a given list of non-empty tuples.

```
def last(n): return n[-1]

def sort_list_last(tuples):
  return sorted(tuples, key=last)

print(sort_list_last([(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]))
```

```
[(2, 1), (1, 2), (2, 3), (4, 4), (2, 5)]
```

Count number of strings

Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.

```
def match_words(words):
  ctr = 0

  for word in words:
    if len(word) > 1 and word[0] == word[-1]:
      ctr += 1
  return ctr

print(match_words(['abc', 'xyz', 'aba', '1221']))
```

```
2
```

Insert string at beginning

Write a Python program to insert a given string at the beginning of all items in a list.

```
num = [1,2,3,4]
print(['emp{0}'.format(i) for i in  num])
```

```
['emp1', 'emp2', 'emp3', 'emp4']
```

Move all zeros to end

Write a Python program to move all zero digits to end of a given list of numbers.

```
# initializing a list
numbers = [1, 3, 0, 4, 0, 5, 6, 0, 7]
# moving all the zeroes to end
new_list = [num for num in numbers if num != 0] + [num for num in numbers if num == 0]
# printing the new list
print(new_list)
[1, 3, 4, 5, 6, 7, 0, 0, 0]
```

```
[1, 3, 4, 5, 6, 7, 0, 0, 0]
[1, 3, 4, 5, 6, 7, 0, 0, 0]
```

Remove duplicates from list of lists

Write a Python program to remove duplicates from a list of lists.

```
import itertools
num = [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]
print("Original List", num)
num.sort()
new_num = list(num for num,_ in itertools.groupby(num))
print("New List", new_num)
```

```
Original List [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]
New List [[10, 20], [30, 56, 25], [33], [40]]
```

```
lst = eval(input())
lst.sort()
new_num = list(lst for lst,_ in itertools.groupby(lst))
a=new_num.pop()
new_num.insert(1,a)
print(new_num)
```

```
[[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]
[[10, 20], [40], [30, 56, 25], [33]]
```

Calculations within list

Write a Python program to round every number of a given list of numbers and print the total sum multiplied by the length of the list.

```
nums = [22.4, 4.0, -16.22, -9.10, 11.00, -12.22, 14.20, -5.20, 17.50]
print("Original list: ", nums)
print("Result:")
```

```
lenght=len(nums)
print(sum(list(map(round,nums))* lenght))
```

Write a Python program to construct the following pattern, using a nested for loop.

```
n=int(input())
for i in range(n):
    for j in range(i):
        print ('*.', end="")
    print('')

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

```
5

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

Print 1 and 0 in Alternative Columns

Write a Python Program to Print 1 and 0 in Alternative Columns.

```
rows = int(input("Please Enter the total Number of Rows  : "))
columns = int(input("Please Enter the total Number of Columns  : "))

print("Print Number Pattern - 1 and 0 in alternative Columns")
i = 1
while(i <= rows):
    j = 1
    while(j <= columns):
        if(j % 2 != 0):
            print('1', end = '.')
        else:
            print('0', end = '.')
        j = j + 1
    i = i + 1
    print()
```

```
Please Enter the total Number of Rows  : 4
Please Enter the total Number of Columns  : 3
Print Number Pattern - 1 and 0 in alternative Columns
1.0.1.
1.0.1.
1.0.1.
1.0.1.
```

Print hollow pattern

Write a Python Program to Print Hollow Rectangle Star Pattern

```
row = 5
col = 5

print("Hollow rectangular pattern is: ")
for i in range(1,row+1):
    for j in range(1,col+1):
        if i==1 or i==row or j==1 or j==col:
            print("0.", end="")
        else:
            print("  ", end="")
    print()
```

```
Hollow rectangular pattern is:
0.0.0.0.0.
0.      0.
0.      0.
0.      0.
0.0.0.0.0.
```

Enter * between identical chars in string

Write a Python Program to Enter * Between two Identical Characters in a String.

```
gvn_str = input()
new_str = ""
itr = 0
for itr in range(0, len(gvn_str)-1):
  new_str = new_str + gvn_str[itr]
  if(gvn_str[itr] == gvn_str[itr+1]):
    new_str += '*'
print(new_str)
```

```
AlmaBetterr
AlmaBet*ter*
```

Reorder list acc. to given index

Write a Python Program to Reorder a List According to Given Indexes.

```
def sort_by_indexes(lst, indexes, reverse=False):
  return [val for (_, val) in sorted(zip(indexes, lst), key=lambda x: \
          x[0], reverse=reverse)]
lst= eval(input())
index= eval(input())
print('The given list after reordering according to the given indexes list :')
print(sort_by_indexes(lst,index))
```

```
print('The Index list after reordering:')
print(sorted(index))
```

```
10, 20, 30, 40, 50, 60
4, 3, 5, 0, 1, 2
The given list after reordering according to the given indexes list :
[40, 50, 60, 20, 10, 30]
The Index list after reordering:
[0, 1, 2, 3, 4, 5]
```

Count number of duplicates Medium Write a program in Python to count number of duplicates in an array having multiple duplicates.

For Example

Number of elements= 5

Enter the element between 0 to 4: 2

Enter the element between 0 to 4: 2

Enter the element between 0 to 4: 3

Enter the element between 0 to 4: 2

Enter the element between 0 to 4: 2

Output: 2 is repeated 4 times

```
num_of_elements = int(input())
a=int(input())
b=int(input())
c=int(input())
d=int(input())
e=int(input())
arr=[a,b,c,d,e]
for i in range(0, len(arr)):
  for j in range(i+1, len(arr)):
    if(arr[i] == arr[j]):
        print(arr[j])
```

```
5
2
2
3
2
2
2
2
2
2
2
2
```

```
def countX(lst, x):
    count = 0
    for ele in lst:
        if (ele == x):
            count = count + 1
```

```
    return count


# Driver Code
lst = [8, 6, 8, 10, 8, 20, 10, 8, 8]
x = 8
print('{} has occurred {} times'.format(x,countX(lst, x)))
```

```
8 has occurred 5 times
```

Remove duplicate elements

Write a program in Python to remove duplicate elements from array.

```
import array
arr, count = [],[]
n = int(input("enter size of array : "))
for x in range(n):
    count.append(0)
    x=int(input("enter element of array : "))
    arr.append(x)
print("Array elements after removing duplicates")
for x in range(n):
    count[arr[x]]=count[arr[x]]+1
    if count[arr[x]]==1:
        print(arr[x])
```

```
enter size of array : 5
enter element of array : 4
enter element of array : 2
enter element of array : 4
enter element of array : 3
enter element of array : 2
Array elements after removing duplicates
4
2
3
```

String with given prefix

Write a Python program to find the strings in a given list, starting with a given prefix.

```
def test(strs, prefix):
     return [s for s in strs if s.startswith(prefix)]
strs =  ['cat', 'car', 'fear', 'center']
prefix = "ca"
print("Original strings:")
print(strs)
print("Starting prefix:", prefix)
print("Strings in the said list starting with a given prefix:")
print(test(strs, prefix))
```

```
Original strings:
['cat', 'car', 'fear', 'center']
Starting prefix: ca
```

```
Strings in the said list starting with a given prefix:
['cat', 'car']
```

Monotonic sequence

Write a Python program to determine the direction ('increasing' or 'decreasing') of monotonic sequence numbers.

```
def test(nums):
    return "Increasing." if all(nums[i] < nums[i + 1] for i in range(len(nums) - 1)) else \
        "Decreasing." if all(nums[i + 1] < nums[i] for i in range(len(nums) - 1)) else \
        "Not a monotonic sequence!"
nums = [1,2,3,4,5,6]
print("Original list:")
print(nums)
print("Check the direction ('increasing' or 'decreasing') of the said list:")
print(test(nums))
nums = [6,5,4,3,2,1]
print("\nOriginal list:")
print(nums)
print("Check the direction ('increasing' or 'decreasing') of the said list:")
print(test(nums))
nums = [19,19,5,5,5,5,5]
print("\nOriginal list:")
print(nums)
print("Check the direction ('increasing' or 'decreasing') of the said list:")
print(test(nums))
```

```
Original list:
[1, 2, 3, 4, 5, 6]
Check the direction ('increasing' or 'decreasing') of the said list:
Increasing.

Original list:
[6, 5, 4, 3, 2, 1]
Check the direction ('increasing' or 'decreasing') of the said list:
Decreasing.

Original list:
[19, 19, 5, 5, 5, 5, 5]
Check the direction ('increasing' or 'decreasing') of the said list:
Not a monotonic sequence!
```

Convert the average to binary

Write a Python program to calculate the average of the numbers a through b (b not included) rounded to nearest integer, in binary (or -1 if there are no such numbers).

```
def test(a,b):
    r = range(a, b)
    if len(r) == 0:
        return "-1"
    return bin(round(sum(r) / len(r)))
a = int(input())
```

```
b = int(input())
print(test(a, b))
```

```
4
7
0b101
```

Unhappy string

Write a Python program to find two indices making a given string unhappy.

Note- A string is happy if every three consecutive characters are distinct.

```
def test(s):
    for i in range(len(s) - 2):
        if s[i] == s[i + 1]:
            return [i, i + 1]
        if s[i] == s[i + 2]:
            return [i, i + 2]

strs = "Python"
print("Original string:",strs)
print("Find two indices making the said string unhappy!")
print(test(strs))
```

```
Original string: Python
Find two indices making the said string unhappy!
None
```

```
def test(s):
    for i in range(len(s) - 2):
        if s[i] == s[i + 1]:
            return [i, i + 1]
        if s[i] == s[i + 2]:
            return [i, i + 2]
n =input()
print(test(n))
```

```
unhappy
[4, 5]
```

Integer start or ends with 2

Write a Python program to find all n-digit integers that start or end with 2.

```
def test(n):
    ans = []
    for i in range(10 ** (n - 1), 10 ** n):
        assert len(str(i)) == n
        if str(i).startswith("2") or str(i).endswith("2"):
            ans.append(i)
    return ans
n = int(input())
print(test(n))
```

```
1
[2]
```

Inject a number

Given a list of numbers and a number to inject, write a Python program to create a list containing that number in between each pair of adjacent numbers.

```
def test(nums, sep):
    ans = [sep] * (2 * len(nums) - 1)
    ans[::2] = nums
    return ans
lst=eval(input())
n = int(input())
print(test(lst,n))
```

```
[12, -7, 3, -89, 14, 88, -78, -1, 2, 7]
6
[12, 6, -7, 6, 3, 6, -89, 6, 14, 6, 88, 6, -78, 6, -1, 6, 2, 6, 7]
```

Vowel between consonants Medium Write a Python program to find a substring in a given string which contains a vowel between two consonants.

A consonant is a speech sound that is not a vowel. It also refers to letters of the alphabet that represent those sounds: Z, B, T, G, and H are all consonants. Consonants are all the non-vowel sounds, or their corresponding letters: A, E, I, O, U and sometimes Y are not consonants.

Example 1: Input: Hello

Output: Hel

```
def test(s):
    cons = "bcdfghjklmnpqrstvwxz"
    vows = "aeiou"
    return next(s[i - 1:i + 2] for i in range(1, len(s) - 1)
                if s[i].lower() in vows and s[i - 1].lower() in cons and s[i + 1].lower() in cons)

n = input()
print(test(n))
```

```
hello
hel
```

Three numbers with sum 0

Write a Python program to find three numbers from an array such that the sum of three numbers equal to zero.

```
def three_Sum(num):
    if len(num)<3: return []
    num.sort()
    result=[]
    for i in range(len(num)-2):
        left=i+1
        right=len(num)-1
        if i!=0 and num[i]==num[i-1]:continue
        while left<right:
            if num[left]+num[right]==-num[i]:
                result.append([num[i],num[left],num[right]])
                left=left+1
                right=right-1
                while num[left]==num[left-1] and left<right:left=left+1
                while num[right]==num[right+1] and left<right: right=right-1
            elif num[left]+num[right]<-num[i]:
                left=left+1
            else:
                right=right-1
    return result
n = eval(input())
print(three_Sum(n))
```

```
[-1,0,1,2,-1,-4]
[[-1, -1, 2], [-1, 0, 1]]
```

Indices of nos with sum 0

Write a Python program to find the indices of three numbers that sum to 0 in a given list of numbers.

```
def test(nums):
    inv = {n: i for i, n in enumerate(nums)}  # note that later duplicates will override earlier entries
    for i, n in enumerate(nums):
        if inv[n] == i:
            del inv[n]
        if any((-m - n) in inv for m in nums[:i]):  # found solution!
            j, m = next((j, m) for j, m in enumerate(nums) if (-m - n) in inv)
            k = inv[-m - n]
            return sorted([i, j, k])
n = eval(input())
print(test(n))
```

```
[12, -7, 3, -89, 14, 4, -78, -1, 2, 7]
[1, 2, 5]
```

Most Common Grade Easy Marty Smart, a Math Teacher at Harwood High, has amassed his student's grades for the semester and has come to you to write a code in python to determine the most common grade.

Example 1: Input: students = {'Marc': 99, 'Amie': 76, 'Jonny': 98, 'Anne': 99, 'Andy': 77, 'Elli': 98, 'Acer': 67, 'Joan': 61, 'Mike': 54, 'Anna': 76, 'Bobi': 67, 'Kate': 99, 'Todd': 98, 'Emma': 49, 'Stan': 76, 'Harv': 99, 'Ward': 67, 'Hank': 54, 'Wendy': 98, 'Sven': 100}

Output: 99

```
from collections import Counter
students = {'Marc': 99, 'Amie': 76, 'Jonny': 98, 'Anne': 99, 'Andy': 77, 'Elli': 98, 'Acer': 67, 'Joan': 61, 'Mike': 54, 'Anna': 76, 'Bobi': 67, 'Kate': 99, 'Todd':
value, count = Counter(students.values()).most_common(1)[0]
print(value)
```

```
99
```

Write a code in python to slice the string from rear end till 5th character from the last.

```
inputt='ALMABETTER'
print(inputt[:4:-1])
```

```
RETTE
```

Write a code in python to reverse the following string using for loop

```
def reverse(s):
    str = ""
    for i in s:
        str = i + str
    return str
s="ALMABETTER"
print(reverse(s))
```

```
RETTEBAMLA
```

Even and Odd characters

Write a python program to print the even and odd index characters of a string.

```
str="ALMABETTER"
even_chars=[]
odd_chars=[]
for i in range(len(str)):
    # check if index is divisible by 2
    if i % 2 != 0:
        even_chars.append(str[i])
    else:
       odd_chars.append(str[i])
print('Odd characters: {}'.format(odd_chars))
print('Even characters: {}'.format(even_chars))
```

```
Odd characters: ['A', 'M', 'B', 'T', 'E']
Even characters: ['L', 'A', 'E', 'T', 'R']
```

Print even length words

Write a Python program to print even length words in a string using list comprehension

```
n="I am working at Almabetter"
#splitting the words in a given string
s=n.split(" ")
l=list(filter(lambda x: (len(x)%2==0),s))
print(l)
```

```
['am', 'at', 'Almabetter']
```

Concatenate reverse lists

Write a python program to divide a list in halves, reverse each halve and concatenate both.

```
list1 = [1,2,3,4,5,6,7,8]


#initialize the middle index with the length of first half
middle_index=4

#Split the list from starting index upto middle index in first half
first_half=list1[:middle_index]

#Split the list from middle index index upto the last index in second half
sec_half=list1[middle_index:]

#printing original lists and two halves
print(first_half[::-1]+sec_half[::-1])
```

```
[4, 3, 2, 1, 8, 7, 6, 5]
```

All indices

Write a python program by using the enumerate() function to get all the indexes for the integer 20 in the “nums” list.

```
nums = [10,15,20,25,30,14,20,19,34,14,20,12]
for i in enumerate(nums):
    if i[1] == 20:
        print(i[0],end=",")
```

```
2,6,10,
```

Fibonacci Series

Write a python program to print fibonacci series upto 15 terms in python.

```
num = 15
n1, n2 = 0, 1
```

```
print("Fibonacci Series:", n1, n2, end=",")
for i in range(2, num):
    n3 = n1 + n2
    n1 = n2
    n2 = n3
    print(n3, end=",")

print()
```

```
 Fibonacci Series: 0 1,1,2,3,5,8,13,21,34,55,89,144,233,377,
```

Convert numbers to words

Write a Python program to get the single digits in numbers sorted backwards and converted to English words.

```
def test(nums):
    digits = {"zero": None,
              "one": 1,
              "two": 2,
              "three": 3,
              "four": 4,
              "five": 5,
              "six": 6,
              "seven": 7,
              "eight": 8,
              "nine": 9}
    digits_backwards = {digits[k]: k for k in digits}
    digits = [digits[s] for s in digits]
    li = [digits[n] for n in nums if n in digits]
    return [digits_backwards[n] for n in sorted(li, reverse=True)]
n = eval(input())
print(test(n))
```

```
 [1, 3, 4, 5, 11]
 ['five', 'four', 'three', 'one']
```

Check extension

A valid filename should end in .txt, .exe, .jpg, .png, or .dll, and should have at most three digits, no additional periods.

Write a Python program to create a list of True/False that determine whether candidate filename is valid or not.

```
def test(file_names):
         return ["Yes" if
            f.split(".")[1:] in [['txt'], ['png'], ['dll'], ['exe'], ['jpg']] and f[0].isalpha() and sum(c.isdigit() for c in f) < 4
            else "No"
            for f in file_names]
n=eval(input())
print(test(n))
```

```
 ['abc.txt', 'windows.dll', 'tiger.png', 'rose.jpg', 'test.py', 'win32.exe']
 ['Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes']
```

Compute sum Medium Surprisingly there are only three numbers that can be written as the sum of fourth powers of their digits:

1634 = 1^4 + 6^4 + 3^4 + 4^4

8208 = 8^4 + 2^4 + 0^4 + 8^4

9474 = 9^4 + 4^4 + 7^4 + 4^4

As 1 = 1^4 is not a sum it is not included.

The sum of these numbers is 1634 + 8208 + 9474 = 19316.

Write a Python program to find the sum of all the numbers (within certain range) that can be written as the sum of fifth powers of their digits.

```
def compute():
  result = sum(i for i in range(2, 1000000) if i == fifthPower_digitSum(i))
  return str(result)

def fifthPower_digitSum(n):
  return sum(int(x)**5 for x in str(n))

print(compute())
```

```
443839
```

Find vowels from each word

Write a Python program to find the vowels from each of the original texts (y counts as a vowel at the end of the word) from a given list of strings.

```
def test(strs):
        return ["".join(c for c in text if c.lower() in "aeiou") + (text[-1] if text[-1].lower() == "y" else "")
            for text in strs]
n = eval(input())
print(test(n))
```

```
['w3resource', 'Python', 'Java', 'C++']
['eoue', 'o', 'aa', '']
```

Find index and sum Medium Write a Python program to find the index of the largest prime in the list and the sum of its digits.

```
def test(nums):
    n, i = max((n, i) for i, n in enumerate(nums) if is_prime(n))
    return [i, sum(int(c) for c in str(n))]
def is_prime(n):
    return n > 1 and all(n % j for j in range(2, int(n ** 0.5) + 1))
n = eval(input())
print("Index of the largest prime in the said list and the sum of its digits:")
print(test(n))
```

```
[3, 7, 4]
Index of the largest prime in the said list and the sum of its digits:
```

```
[1, 7]
```

Closest distinct numbers

Write a Python program to find the two closest distinct numbers in a given a list of numbers.

```
def test(nums):
    s = sorted(set(nums))
    return min([[a, b] for a, b in zip(s, s[1:])], key=lambda x: x[1] - x[0])
lst = eval(input())
print(test(lst))
```

```
[1.3, 5.24, 0.89, 21.0, 5.27, 1.3]
[5.24, 5.27]
```

Find all 5's in integers

Write a Python program to find all 5's in integers less than n that are divisible by 9 or 15.

```
def test(n):
    return [[i,j] for i in range(n) for j in range(len(str(i))) if str(i)[j] == '5' and (i%15==0 or i%9==0)]
n = int(input())
print(test(n))
```

```
50
[[15, 1], [45, 1]]
```

Shift decimal digits

Write a Python program to shift the decimal digits n places to the left, wrapping the extra digits around. If shift > the number of digits of n, reverse the string.

```
def test(n, shift):
    s = str(n)
    if shift > len(s):
        return s[::-1]
    return s[shift:] + s[:shift]
n = int(input())
shift= int(input())
print(test(n, shift))
```

```
12345
1
23451
```

Sum of even elements

Write a Python program to find the sum of the even elements that are at odd indices.

```
def test(nums):
    return sum(i for i in nums[1::2] if i % 2 == 0)
```

```
n = eval(input())
print(test(n))
```

```
[1, 2, 8, 3, 9, 4]
6
```

Factorial using recursive method

Write a python program to find the factorial of the number using recursion method.

```
def recur_factorial(n):
   if n == 1:
       return n
   else:
       return n*recur_factorial(n-1)

num = int(input())
if num < 0:
   print("Factorial can't be calculated for negative number")
elif num == 0:
   print("Factorial of 0 is 1")
else:
   print("Factorial of", num, "is", recur_factorial(num))
```

```
5
Factorial of 5 is 120
```

Find alphanumeric words

Write a program to find words with both alphabets and numbers from an input string.

```
string='Emma25 is Data scientist50 and AI Expert'
res = []
temp = string.split()
for idx in temp:
    if any(chr.isalpha() for chr in idx) and any(chr.isdigit() for chr in idx):
        res.append(idx)

# printing result
print("Displaying words with alphabets and numbers")
print(str(res))
```

```
Displaying words with alphabets and numbers
['Emma25', 'scientist50']
```

Pack consecutive duplicates

Write a Python program to pack consecutive duplicates of a given list elements into sublists.

```
from itertools import groupby
def pack_consecutive_duplicates(l_nums):
    return [list(group) for key, group in groupby(l_nums)]
```

```
l=eval(input())
print(pack_consecutive_duplicates(l))
```

```
[0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9]
[[0, 0], [1], [2], [3], [4, 4], [5], [6, 6, 6], [7], [8], [9]]
```

Print Combo Pattern

Write a Python Program to Print Pattern with a Combination of Numbers and Stars.

```
row = 8
for i in range(0, row):
    c = 1
    print(c, end='')
    for j in range(row - i - 1, 0, -1):
        print('*', end='')
        c = c + 1
        print(c, end=' ')
    print('')
```

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

Compress string as per rules Hard When character are consecutive in a string , it is possible to shorten the character string by replacing the character with a certain rule. For example, in the case of the character string YYYYY, if it is expressed as # 5 Y, it is compressed by one character.

Write a Python program to restore the original string by entering the compressed string with this rule. However, the # character does not appear in the restored character string.

Note: The original sentences are uppercase letters, lowercase letters, numbers, symbols, less than 100 letters, and consecutive letters are not more than 9 letters.

```
def restore_original_str(a1):
  result = ""
  ind = 0
  end = len(a1)
  while ind < end:
    if a1[ind] == "#":
      result += a1[ind + 2] * int(a1[ind + 1])
      ind += 3
    else:
      result += a1[ind]
      ind += 1
  return result
```

Solve Equations

Write a Python program which solve the equation:

ax+by=c

dx+ey=f

Print the values of x, y where a, b, c, d, e and f are given.

```
a, b, c, d, e, f = map(float, input().split())
n = a*e - b*d
print("Values of x and y:")
if n != 0:
    x = (c*e - b*f) / n
    y = (a*f - c*d) / n
    print('{:.3f} {:.3f}'.format(x+0, y+0))
```

```
5 8 6 7 9 4
Values of x and y:
-2.000 2.000
```

Find list with four distinct values

Write a Python program to find list integers containing exactly four distinct values, such that no integer repeats twice consecutively among the first twenty entries.

```
def test(nums):
    return all([nums[i] != nums[i + 1] for i in range(len(nums)-1)]) and len(set(nums)) == 4
l=eval(input())
print(test(l))
```

```
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
True
```

Max of first ith elements

Write a Python program to create a list whose ith element is the maximum of the first i elements from a input list.

```
def test(nums):
    return [max(nums[:i]) for i in range(1, len(nums) + 1)]
n = eval(input())
print(test(n))
```

```
[0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -4, -15, 0]
[0, 0, 3, 8, 8, 9, 9, 14, 14, 14, 14, 14, 14, 17, 41, 41, 41, 41, 41, 41]
```

Find rotating string

Write a Python Program to Determine Whether one String is a Rotation of Another

```
def checkString(s1, s2, indexFound, size):
```

```
def checkString(s1, s2, indexFound, Size):
    for i in range(Size):


        if(s1[i] != s2[(indexFound + i) % Size]):
            return False
    return True

s1 = input()
s2 = input()

if(len(s1) != len(s2)):
    print("s2 is not a rotation on s1")

else:
    indexes = []
    Size = len(s1)
    firstChar = s1[0]
    for i in range(Size):
        if(s2[i] == firstChar):
            indexes.append(i)

    isRotation = False

    for idx in indexes:

        isRotation = checkString(s1, s2, idx, Size)

        if(isRotation):
            break
    if(isRotation):
        print("The second string is not the rotation of the first string")
    else:
        print("The second string is the rotation of the first string")
```

```
"btechgeeks"
"geeksbtech"
The second string is the rotation of the first string
```

Python Program to Check Prime Number

```
num = 407

# To 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
```

✓ 0s completed at 9:08 PM