|  |  |
| --- | --- |
| def guess\_number(guess):      if guess == 42:          return 'You win!'      return 'Try again!' | y = 61 def guess\_number (x):  if x == y :  return 'You win!'  return 'Try again!' x = 42 guess\_number(x) |

Реализуйте функцию guess\_number(), которая принимает число и проверяет, равно ли число заданному (пусть это будет 42). Если равно, то функция должна вернуть строку 'You win!', в противном случае нужно вернуть строку 'Try again!'.

guess\_number(42) # 'You win!'

guess\_number(61) # 'Try again!'

|  |  |
| --- | --- |
| def get\_type\_of\_sentence(sentence):  last\_char = sentence[-1]  if last\_char == '?':  sentence\_type = 'question'  else:  sentence\_type = 'normal' | def guess\_number(guess):      if guess == 42:          return 'You win!'      return 'Try again!' |

Не совсем равное – разница возврат и присвоение

#Programist  
#misha, [29.03.2022 15: 47] Обработка линейного массива как двумерного  
r = 24  
w = 4  
h = r / w  
x = 0  
y = 0  
for i in range(0, r):  
 if i % w == 0:  
 x = 0  
 if i % h == 0:  
 y = 0  
  
 x = x + 1  
 y = y + 1  
 print(x, y)

url = input()  
def normalize\_url(url):  
 pref = 'https://'  
 if url[:8] == 'https://':  
 return url  
 else:  
 if url[:7] == 'http://':  
 return pref + url[7:]  
 else:  
 return pref + url

Реализуйте функцию normalize\_url(), которая выполняет нормализацию данных. Она принимает адрес сайта и возвращает его с https:// в начале. Функция принимает адреса в виде АДРЕС или http://АДРЕС, но всегда возвращает адрес в виде https://АДРЕС. На вход функции также может поступить адрес в уже нормализованном виде https://АДРЕС, в этом случае ничего менять не надо.

Имитация загрузки 2 варианта

1

1print ('имитация загрузки ')  
import time, sys  
x = int(input('время загрузки : '))  
stars = int(input('количество звезд: '))  
for i in range (stars):  
 print('\*', end=' ')  
 sys.stdout.flush()  
 time.sleep(x / stars)

2

print ('иммитация загрузки ')  
import time, sys  
total = int(input('Время загрузки (сек):'))  
count = 15  
for i in range(count):  
 print('\*', end=' ')  
 sys.stdout.flush()  
 time.sleep(total/count)  
print('\n')

На электронной карте Вестероса, которую реализовал Сэм, союзники Старков отображены зелёным кружком, враги — красным, а нейтральные семьи — серым.

Напишите для Сэма функцию who\_is\_this\_house\_to\_starks(), которая принимает на вход фамилию семьи и возвращает одно из трёх значений: 'friend', 'enemy', 'neutral'.

Правила определения:

* Друзья ('friend'): 'Karstark', 'Tully'
* Враги ('enemy'): 'Lannister', 'Frey'
* Любые другие семьи считаются нейтральными ('neutral')
* def who\_is\_this\_house\_to\_starks(for\_your\_house):  
   if for\_your\_house == 'Karstark' or for\_your\_house == 'Tully':  
   return 'friend'  
   elif for\_your\_house == 'Lannister' or for\_your\_house == 'Frey':  
   return 'enemy'  
   return 'neutral'  
  print(who\_is\_this\_house\_to\_starks('Karstark'))  
  print(who\_is\_this\_house\_to\_starks('Frey'))  
  print(who\_is\_this\_house\_to\_starks('Joar'))  
  print(who\_is\_this\_house\_to\_starks('Ivanov'))

Реализуйте функцию flip\_flop(), которая принимает на вход строку и, если эта строка равна 'flip', возвращает строку 'flop'. В противном случае функция должна вернуть 'flip

def flip\_flop(x):  
 return 'flop' if x == 'flip' else 'flip'  
x = input('input flop or flip: ')  
print(flip\_flop(x))

Модифицируйте функцию print\_numbers() так, чтобы она выводила числа в обратном порядке. Для этого нужно идти от верхней границы к нижней. То есть счётчик должен быть инициализирован максимальным значением, а в теле цикла его нужно уменьшать до нижней границы.

def print\_numbers(Last\_N) :  
 i = Last\_N  
 while i > 0:  
 print(i)  
 i = i - 1  
 print ('finished')  
Last\_N = int(input('введи число: '))  
print(print\_numbers(Last\_N))

Реализуйте функцию multiply\_numbers\_from\_range(), которая перемножает числа в указанном диапазоне включая границы диапазона. Пример вызова:

multiply\_numbers\_from\_range(1, 5) # 1 \* 2 \* 3 \* 4 \* 5 = 120

multiply\_numbers\_from\_range(2, 3) # 2 \* 3 = 6

multiply\_numbers\_from\_range(6, 6) # 6

def multiply\_numbers\_from\_range (st, fin):  
 i = st  
 multip = 1  
 while i <= fin:  
 multip = multip \* i  
 i = i + 1  
 print (multip)  
 return (multip)  
multiply\_numbers\_from\_range (1,3)

Реализуйте функцию join\_numbers\_from\_range(), которая объединяет все числа из диапазона в строку:

join\_numbers\_from\_range(1, 1) # '1'

join\_numbers\_from\_range(2, 3) # '23'

join\_numbers\_from\_range(5, 10) # '5678910'

def join\_numbers\_from\_range(st, fin):  
 i = st  
 result = ''  
 while i <= fin:  
 re

sult = result + str(i)  
 i = i + 1  
 print(result)  
 return result  
print(join\_numbers\_from\_range(1, 8))

Реализуйте функцию print\_reversed\_word\_by\_symbol(), которая печатает переданное слово посимвольно, как в примере из теории, но делает это в обратном порядке.

|  |  |
| --- | --- |
| def **print\_reversed\_word\_by\_symbol**(**word**):  i = len(**word**) - 1  **while** i >= 0:  print(**word**[i])  i = i - 1 | def print\_reversed\_word\_by\_symbol(**name**):  2 i = 0  3 i\_name = name[::-1]  4 while i < len(**name**):  5 print(**i\_name**[i])  6 i = i + 1 |

функцию, которая считает, сколько раз входит буква в предложение.

def count\_chars(string, char):

index = 0

count = 0

while index < len(string):

if string[index] == char:

# Считаем только подходящие символы

count = count + 1

# Счётчик увеличивается в любом случае

index = index + 1

return count

функцию, которая считает, сколько раз входит буква в предложение. Пример её работы:  Реализуйте вариант этой же функции, так чтобы регистр букв был не важен:

|  |  |
| --- | --- |
| def count\_chars(string, char):      string = string.lower()      char = char.lower()      index = 0      count = 0      while index < len(string):          if string[index] == char:              # Считаем только подходящие символы              count = count + 1          # Счётчик увеличивается в любом случае          index = index + 1      return count | def count\_chars(string, char):  2 string = string.lower()  3 char = char.lower()  4 **index** = 0  5 **count** = 0  6 **while** **index** < len(string):  7 **if** string[**index**] == char:  8 # Считаем только подходящие символы  9 **count** = **count** + 1  10 # Счётчик увеличивается в любом случае  11 **index** = **index** + 1  12 **return** **count** |

Реализуйте функцию my\_substr(), которая извлекает из строки подстроку указанной длины. Она принимает на вход два аргумента: строку и длину, и возвращает подстроку, начиная с первого символа:

|  |  |
| --- | --- |
| def x(y, m, n):  i = m  resultat = ''  while i <= n:  resultat = resultat + y[i]  i = i+1  return resultat print (x('repachka', 2, 5) | def x(text, st, en):  resultat = ''  for i in text[st:en]:  resultat = resultat + i  return resultat print (x('repachka', 2, 4)) |

Реализуйте функцию-предикат is\_arguments\_for\_substr\_correct(), которая принимает три аргумента:

1. строку;
2. индекс, с которого начинать извлечение;
3. длину извлекаемой подстроки.

Функция возвращает False, если хотя бы одно из условий истинно:

* Отрицательная длина извлекаемой подстроки.
* Отрицательный заданный индекс.
* Заданный индекс выходит за границу всей строки.
* Длина подстроки в сумме с заданным индексом выходит за границу всей строки.

|  |  |
| --- | --- |
| * def is\_arguments\_for\_substr\_correct(string, **index**, length): * **if** **index** < 0: * **return** False * elif length < 0: * **return** False * elif **index** > len(**string**) - 1: * **return** False * elif **index** + length > len(**string**): * **return** False * **return** True | def x(text, st, en):  st = int(st)  en = int(en)  resultat = ''  if 0 > st or st >= len(text):  print('error st')  else:  if 0 < en or en <= len(text):  print('error en')  else:  for i in text[st:en]:  resultat = resultat + i  return resultat print (x('text',-1 , 5)) |
| def x(text, st, en):  st = int(st)  en = int(en)  resultat = ''  for i in text[st:en]:  if 0 > st or st >= len(text):  print('error st')  break  if 0 < en or en <= len(text):  print ('error en')  break  resultat = resultat + i  return resultat print (x('text',-1 , 5)) | def substr(text, start, end):  length = len(text)  if start >= 0 and end < length:  return text[start:end]  else:  return f'error: [{text}]'  print(substr("Hello world", 0, 6))  print(substr("Hello world", -1, 3))  print(substr("Hello world", 3, 5))  print(substr("Hello world", -15, 15))  print(substr("Hello world", 7, 10))  print(substr("Hello world", 0, 15)) |

* Реализуйте функцию filter\_string(), принимающую на вход строку и символ, и возвращающую новую строку, в которой удален переданный символ во всех его позициях.

|  |  |
| --- | --- |
| * def filter\_string(text, i): * return text.replace(i, '') * #return text\_repl * print(filter\_string('spacep', 'p')) | def filter\_string(text, char):  index = 0  result = ''  while index < len(text):  current\_char = text[index]  if current\_char != char:  result += current\_char  index += 1  return resul |
| print('spacep'.replace('p', '')) | s = 'spacep' print(s.replace('p', '')) |

Рассмотрим простой алгоритм проверки простоты числа. Будем делить искомое число x на все числа из диапазона от двух до x - 1 и смотреть остаток. Если в этом диапазоне не найден делитель, который делит число x без остатка, значит перед нами простое число.

Если задуматься, то можно заметить, что достаточно проверять числа не до x - 1, а до половины числа. Например, 11 не делится на 2, 3, 4, 5. Но и дальше гарантированно не будет делиться на числа больше своей половины. Значит, можно провести небольшую оптимизацию и проверять деление только до x / 2.

Алгоритм построен таким образом, о если во время последовательного деления на числа до x / 2 находится хоть одно, которое делит без остатка, то переданный аргумент — не простое число, а значит дальнейшие вычисления не имеют смысла. В этом месте стоит возврат False.

И только если цикл отработал целиком, можно сделать вывод, что число — простое, так как не было найдено ни одного числа, которое делит число без остатка.

def is\_prime(number):

if number < 2:

return False

divider = 2

while divider <= number / 2:

if number % divider == 0:

return False

divider += 1

return True

print(is\_prime(1)) # => False

print(is\_prime(2)) # => True

print(is\_prime(3)) # => True

Реализуйте функцию is\_contains\_char(), которая проверяет с учётом регистра, содержит ли строка указанную букву. Функция принимает два параметра:

* Строка
* Буква для поиска

print(is\_prime(4)) # => False

(#без учета регистра)

def is\_contains\_char(text, letter):  
 text = text.lower()  
 letter = letter.lower()  
 index = 0  
 while index < len(text):  
 if text[index] == letter:  
 return True  
 index = index+1  
 return False

В одном из предыдущих уроков мы уже написали функцию filter\_string(). Напомним, она принимает на вход строку и символ и возвращает новую строку, в которой удалён переданный символ во всех его позициях. На этот раз реализуйте эту функцию с помощью цикла for. Дополнительное условие: регистр исключаемого символа не имеет значения.

def filter\_string(text, litter):  
 litter = litter.lower()  
 result = ''  
 for i in range(len(text)):  
 if litter != text.lower()[i]:  
 result += text[i]  
 return result  
print(filter\_string('If I look forward I am win', 'o'))

дуальные выражения ниже с выражение ранж !!!

if (-15 < x and x <= 12) or (14 < x and x < 17) or x >= 19:  
if x in range(-15, 13) or x in range(14, 17) or x >= 19:

Напишите простой калькулятор, который считывает с пользовательского ввода три строки: первое число, второе число и операцию, после чего применяет операцию к введённым числам ("первое число" "операция" "второе число") и выводит результат на экран.

Поддерживаемые операции: +, -, /, \*, mod, pow, div, где  
mod — это взятие остатка от деления,  
pow — возведение в степень,  
div — целочисленное деление.

Если выполняется деление и второе число равно 0, необходимо выводить строку "Деление на 0!".

Обратите внимание, что на вход программе приходят вещественные числа.

print ('Вычисление Х , Y и операция с ними (+, - , mod = %, pow \*\*, div= //,/):')  
x = float(input('Введите Х: ')) # перовое число  
y = float(input('Введите Y: ')) # второе число  
z = input('Операция: ') # операция  
p = ''  
# +, - , mod = %, pow \*\*, div= //,/  
if z == "+":  
 p = x + y  
elif z == "-":  
 p = x - y  
elif z == '\*':  
 p = x \* y  
elif z == 'pow':  
 p = x\*\*y  
elif y == 0:  
 print('На ноль делить нельзя')  
elif z == 'div' and y !=0:  
 p = x // y  
elif z == "/" and y !=0:  
 p = x / y  
elif z == 'mod' and y !=0:  
 p = x % y  
print (p)

# Метод сложения  
def plus(a,b):  
 return a + b  
  
# Метод вычитания  
def minus(a,b):  
 return a - b  
  
'''  
Словарь ключ:значение, где ключ это строковый знак/операция  
А значение - функции выше  
'''  
calc\_dict = {  
 "+": plus,  
 "-": minus  
}  
  
# Получаем все значения  
a = float(input("a: "))  
b = float(input("b: "))  
op\_str = input("operation: ")  
  
'''  
Получаем функцию из словаря по ключу  
То есть, если ключ + то отдаст функцию plus  
Если ключ - то отдаст функцию minus  
'''  
op\_func = calc\_dict[op\_str]  
  
# Применяем функцию, по сути не зная её имени  
result = op\_func(a,b)  
  
print(result)

Сделано через словарь аргументов

calc\_dict = {  
 "+": lambda x,y: x + y,  
 "-": lambda x,y: x - y,  
 "\*": lambda x,y: x \* y,  
 "/": lambda x,y: x / y,  
 "div": lambda x,y: x // y,  
 "pow": lambda x,y: x \*\* y,  
 "mod": lambda x,y: x % y,  
}  
  
a = float(input("a: "))  
b = float(input("b: "))  
  
operation\_str = input("operation: ")  
operation\_func = calc\_dict[operation\_str]  
  
result = operation\_func(a,b)  
  
print(result)

Жители страны Малевии часто экспериментируют с планировкой комнат. Комнаты бывают треугольные, прямоугольные и круглые. Чтобы быстро вычислять жилплощадь, требуется написать программу, на вход которой подаётся тип фигуры комнаты и соответствующие параметры, которая бы выводила площадь получившейся комнаты.  
Для числа π в стране Малевии используют значение 3.14.

Формат ввода, который используют Малевийцы:

import math  
ch = input('выбор фигуры : ')  
# круг==circle , прямоугольник ==rectangle , треугольник == triangle, math.pi - пи 3,14  
if ch == 'круг':  
 r = float(input('value\_r: '))  
 print ('S\_круга: ', math.pi \* (r\*\*2) )  
elif ch == "прямоугольник":  
 a = float(input('value\_a: '))  
 b = float(input('value\_b: '))  
 print ('S\_прямоугольника: ', a\*b)  
elif ch == 'треугольник':  
 a = float(input('value\_a: '))  
 b = float(input('value\_b: '))  
 c = float(input('value\_c: '))  
 p = (a+b+c)/2  
 print ('S\_треугольника: ', (p\*(p-a)\*(p-b)\*(p-c))\*\*(1/2))

Напишите программу, которая получает на вход три целых числа, по одному числу в строке, и выводит на консоль в три строки сначала максимальное, потом минимальное, после чего оставшееся число.

На ввод могут подаваться и повторяющиеся числа.

a = float(input('value1: '))  
b = float(input('value2: '))  
c = float(input('value3: '))  
list = [a, b, c]  
list.sort()  
print(int(list[-1]))  
print(int(list[0]))  
print(int(list[1]))

тоже только колличество сравниваемых аргументок равно N= COUNT

params = []  
count = int(input("count:"))  
for i in range(1,count+1):  
 inp = int(input(f"{i}:"))  
 params.append(inp)  
print(params)  
list = params  
params.sort()  
print(params)  
list.sort()  
print('max', int(list[-1]))  
print('min', int(list[0]))  
#print('middle', int(list[1]))  
print('middle', list[1:-1])

В институте биоинформатики по офису передвигается робот. Недавно студенты из группы программистов написали для него программу, по которой робот, когда заходит в комнату, считает количество программистов в ней и произносит его вслух: "n программистов".

Для того, чтобы это звучало правильно, для каждого n*n* нужно использовать верное окончание слова.

Напишите программу, считывающую с пользовательского ввода целое число n*n* (неотрицательное), выводящее это число в консоль вместе с правильным образом изменённым словом "программист", для того, чтобы робот мог нормально общаться с людьми, например: 1 программист, 2 программиста, 5 программистов.

В комнате может быть очень много программистов. Проверьте, что ваша программа правильно обработает все случаи, как минимум до 1000 человек.

**Дополнительный комментарий к условию:**  
Обратите внимание, что задача не так проста, как кажется на первый взгляд. **Если** ваше **решение не проходит** какой-то **тест**, это значит, что **вы не рассмотрели какой-то из случаев** входных данных (число программистов 0 \le n \le 10000≤*n*≤1000). Обязательно проверяйте свои решения на дополнительных значениях, а не только на тех, что приведены в условии задания.

Так как **задание повышенной сложности**, **вручную код решений проверяться не будет**. Если вы столкнулись с ошибкой в первых четырёх тестах, проверьте, что вы используете только русские символы для ответа. В остальных случаях ищите ошибку в логике работы программы.

count\_input = input('count: ')  
count = int(count\_input[-2:])# только диапозон 11-20  
count\_1 = int(count\_input[-1])  
#print(count, count\_1)  
if count in range (11 ,21) or count\_1 == 0: # 11 <= count and count <= 20 or count\_1 == 0  
 print(count\_input, ': программистов')  
elif count\_1 == 1:  
 print (count\_input,': программист')  
elif count\_1 in range (2, 5): # 2 <= count\_1 and count\_1 <=4  
 print (count\_input, ': программиста')  
elif count\_1 in range (5,10): # 5 <= count\_1 and count\_1 <= 9  
 print(count\_input, ': программистов')

или

def programmers(count\_input):  
 last\_two = int(count\_input[-2:])  
 last\_one = int(count\_input[-1])  
 if last\_two in range(5, 21):  
 return 'программистов'  
 if last\_one == 1:  
 return 'программист'  
 if last\_one in range(2, 5):  
 return 'программиста'  
 if last\_one in range(5, 10) or last\_one == 0:  
 return 'программистов'  
enter = input()  
word = f"{enter} {programmers(enter)}"  
print(word)

Паша очень любит кататься на общественном транспорте, а получая билет, сразу проверяет, счастливый ли ему попался. Билет считается счастливым, если сумма первых трех цифр совпадает с суммой последних трех цифр номера билета.  
  
Однако Паша очень плохо считает в уме, поэтому попросил вас написать программу, которая проверит равенство сумм и выведет "Счастливый", если суммы совпадают, и "Обычный", если суммы различны.  
  
На вход программе подаётся строка из шести цифр.

Выводить нужно только слово "Счастливый" или "Обычный", с большой буквы.

|  |  |
| --- | --- |
| x = input("введи 6 значное число : " ) w\_l = x[0:3] w\_r = x[3:] sum\_wl =int(w\_l[0]) + int(w\_l[1]) + int(w\_l[2]) sum\_wr = int(w\_r[0]) + int(w\_r[1]) + int(w\_r[2]) if sum\_wl == sum\_wr:  print ('Счастливый') else:  print ('Обычный') # sum\_wl = w\_l[3] + w\_l[4] + w\_l[5] # for i in range(word\_left): # sum\_wl = word\_left ( print(sum\_wr) print(sum\_wl) |  |

Вычисление суумы числе на отрезке а до б ( s сумма чисел )

a = int(input('a: '))  
b = int(input('b: '))  
s = 0 # сумма на отрезке а и б  
i = a  
while i <= b:  
 s = s + i # s +=i  
 i +=1  
print(s)

Напишите программу, которая считывает со стандартного ввода целые числа, по одному числу в строке, и после первого введенного нуля выводит сумму полученных на вход чисел.

|  |  |
| --- | --- |
| a = float('-inf') # минус бесконечность sum = 0 #сумма чисел i = 1 while i > a :  x = int(input())  sum = sum + x  if x ==0:  break print (sum) | s = 0 i = 1 while i:  i = int(input())  s += i print(s) |
| a,b = 1,0 while a != 0:  a = int(input())  b += a print(b)  или  summa = 0 i = True while i:  i = int(input())  summa += i print(summa) | n=int(input()) #считываем целое число s=0 #сумма чисел изначально равна нулю while n != 0: #запускаем цикл с условием  s=s+n #прибавляем к сумме введённое число  n=int(input()) #просим ввести число повторно print(s) #выводим сумму |

Напишите программу, которая помогает найти это число.  
Программа должна считывать размеры команд (два положительных целых числа *a* и *b*, каждое число вводится на отдельной строке) и выводить наименьшее число *d*, которое делится на оба этих числа без остатка.

|  |  |
| --- | --- |
| #найти минимальное число на которе делятся а и б без остатка i = 1 a = int(input('число а: ')) b = int(input('число b: ')) sum = 0 while i > 0:  sum = sum + i  if sum % a ==0 and sum % b ==0:  break print(sum) | a, b = int(input()), int(input()) s = a while s % a or s % b:  s += a print(s) |

Для каждого введённого числа проверить:  
если число меньше 10, то пропускаем это число;  
если число больше 100, то прекращаем считывать числа;  
в остальных случаях вывести это число обратно на консоль в отдельной строке.

|  |  |
| --- | --- |
| sum\_s ='' while True:  x = int(input())  if x >= 10 and x <= 100:  sum\_s = f'{sum\_s}\n{str(x)}' # можно заменить на sum\_s += str(x) + '\n' и убрать [-1:]  elif x > 100:  break print(sum\_s[1:]) print(sum\_s[1:].split()) | То же в списках  param = [] while True:  x = int(input())  if x >= 10 and x <= 100:  param.append(x)  elif x > 100:  break print(param) |

param = []  
while True:  
 x = int(input())  
 if x >= 10 and x <= 100:  
 param.append(x)  
 elif x > 100:  
 break  
#for y in param:  
 # print(y)  
print(param)  
#print(sum\_s[1:])  
#print(sum\_s[1:].split())

import sys  
a =int(input())  
b =int(input())  
c =int(input())  
d =int(input())  
if a > b or c > d:  
 print ('ошибка')  
 sys.exit()  
if a >10 or b > 10 or c > 10 or d > 10:  
 print('ошибка')  
 sys.exit()  
print ('','\t', c,'', '\t', d)  
for i in range(a, b+1):  
 for j in range(c, d+1):  
 x = i \* c  
 y = i \* d  
 print(i,'\t',x,'\t',y)

Напишите программу, на вход которой даются четыре числа a, b, c и d, каждое в своей строке. Программа должна вывести фрагмент таблицы умножения для всех чисел отрезка [a; b][*a*;*b*] на все числа отрезка [c;d][*c*;*d*].

Числа a, b, c и d являются натуральными и не превосходят 10, a *a*≤*b*, c *c*≤*d*.

Следуйте формату вывода из примера, для разделения элементов внутри строки используйте '\t' — символ табуляции. Заметьте, что левым столбцом и верхней строкой выводятся **сами числа из заданных отрезков** — заголовочные столбец и строка таблицы.

|  |  |
| --- | --- |
| *''' проверка условий задачи '''* import sys a =int(input()) b =int(input()) c =int(input()) d =int(input()) if a > b or c > d:  print ('ошибка')  sys.exit() if a >10 or b > 10 or c > 10 or d > 10:  print('ошибка')  sys.exit() ''' решение ''' stro1 = '\t ' for k in range (c, d+1):  stro1 += str(k) + '\t' print(stro1) for i in range(a, b+1):  stro = '\t '  for k in range (c, d+1):  stro +=str(k\*i)+'\t'  print(i, stro) | a=int(input()) b=int(input()) c=int(input()) d=int(input())  print('\t', \*range(c, d+1), sep='\t') for i in range(a,b+1):  print(i, \*range(i\*c,(i\*d)+1, i), sep='\t') |
| a,b,c,d = ( int(input()) for i in range(4) ) print('\t', \*range(c, d+1), sep='\t') [ print(\*[i, \*range(i\*c, i\*d+1, i)], sep='\t') for i in range(a, b+1) ] | a, b, c, d = (int(input()) for x in range(4)) print('', \*range(c,d+1), sep='\t') for x in range(a, b+1):  print(x, \*[y\*x for y in range(c, d+1)], sep='\t') |

В приведенном ниже примере среднее арифметическое считается для чисел на отрезке [-5; 12][−5;12]. Всего чисел, делящихся на 33, на этом отрезке 66: -3, 0, 3, 6, 9, 12−3,0,3,6,9,12. Их среднее арифметическое равно 4.54.5

На вход программе подаются интервалы, внутри которых всегда есть хотя бы одно число, которое делится на 33.﻿

|  |  |
| --- | --- |
| в | a = int(input()) b = int(input()) r = range(a,b + 1) r = [i for i in r if i % 3 == 0] print(sum(r)/len(r)) |
| r = [i for i in range(a,b + 1) if i % 3 == 0]  расписал строку  r = [] for i in range(a, b+1):  if i % 3 == 0:  r.append(i) | a,b = int(input()), int(input()) a += -a%3 b -= b%3 print((a+b)/2) |

Узнав, что ДНК не является случайной строкой, только что поступившие в Институт биоинформатики студенты группы информатиков предложили использовать алгоритм сжатия, который сжимает повторяющиеся символы в строке.

Кодирование осуществляется следующим образом:  
s = 'aaaabbсaa' преобразуется в 'a4b2с1a2', то есть группы одинаковых символов исходной строки заменяются на этот символ и количество его повторений в этой позиции строки.

Напишите программу, которая считывает строку, кодирует её предложенным алгоритмом и выводит закодированную последовательность на стандартный вывод. Кодирование должно учитывать регистр символов.

Мишино решение

from itertools import groupby  
  
inp = input()  
groups = [''.join(list(g)) for k, g in groupby(inp)]  
result = ''.join([f'{v[0]}{len(v)}' for v in groups])  
print(result)

Кодирование осуществляется следующим образом:  
s = 'aaaabbсaa' преобразуется в 'a4b2с1a2', то есть группы одинаковых символов исходной строки заменяются на этот символ и количество его повторений в этой позиции строки.

Напишите программу, которая считывает строку, кодирует её предложенным алгоритмом и выводит закодированную последовательность на стандартный вывод. Кодирование должно учитывать регистр символов.

text = 'teetttxt'  
k = 0  
length = len(text)  
result = ''  
  
for i in range(length):  
 k += 1  
 cur = text[i]  
 if i == length - 1:  
 result += cur + str(k)  
 break  
 nex = text[i+1]  
 if cur == nex:  
 pass  
 if cur != nex:  
 result += cur + str(k)  
 k = 0  
print(result)

s = input()  
s1 = ""  
a = 0  
b = s[0]  
for i in s:  
 if i == b:  
 a += 1  
 else:  
 s1 += b + str(a)  
 a = 1  
 b = i  
s1 += b + str(a)  
print(s1)

То действие, которые является особым, т.е. прекращается последовательность - можно обрабатывать в условии. А накручивать счётчик при прохождении символов - это "нормальное" поведение программы, поэтому его можно выполнять без дополнительных условных блоков.  
Верное решение #28646944  
Python 3

genome = input()+' '  
s = 0  
n=genome[0]  
for i in genome:   
 if n!=i:  
 print(n + str(s), end = '')  
 s=0  
 n=i

s = input()  
i = 0 # индекс элемента  
cnt = 1 # счетчик повторений  
result = '' # результат в виде строки  
if len(s) == 1: # если длина строки равна 1:  
 print(s[i] + '1') # то выведи - 'символ1'  
else: # иначе:  
 while i < len(s) - 1: # пока индекс элемента меньше длины строки минус 1, т.е. пока не переберешь все элементы строки:  
 if s[i] == s[i + 1]: # если соседние элементы равны:  
 cnt += 1 # то прибавь к счетчику 1  
 else: # иначе:  
 result += s[i] + str(cnt) # сохрани в переменную результат: результат + символ + количество повторений  
 cnt = 1 # обнули счетчик, т.е. задай начальный параметр  
 i += 1 # увеличивай индекс элемента на 1  
 print(result + s[i] + str(cnt)) # выведи: результат + символ + количество повторений

обратное задание предыдущиму ( дешефрует текст )

|  |  |
| --- | --- |
| shifr ='t1e2t3x1t1' result ='' sh = len(shifr) cur = '' for i in range(0,sh,2):  cur += str(shifr[i])  result +=cur\*(int(shifr[i+1])) print(result) print(cur) | shifr ='t1e2t3x1t1' sh = len(shifr) result = '' for i in range(0,sh,2):  letter = shifr[i]  count = shifr[i+1]  result += letter \* int(count) print(result) |

Напишите программу, на вход которой подается одна строка с целыми числами. Программа должна вывести сумму этих чисел.

Используйте метод split строки

|  |  |
| --- | --- |
| a = [int(i)for i in input().split()] b = 0 for i in a:  b += i  #print(b) print(b) | s = 0 for i in input().split():  s += int(i) print(s) |

Напишите программу, на вход которой подаётся список чисел одной строкой. Программа должна для каждого элемента этого списка вывести сумму двух его соседей. Для элементов списка, являющихся крайними, одним из соседей считается элемент, находящий на противоположном конце этого списка. Например, если на вход подаётся список "1 3 5 6 10", то на выход ожидается список "13 6 9 15 7" (без кавычек).

Если на вход пришло только одно число, надо вывести его же.

Вывод должен содержать одну строку с числами нового списка, разделёнными пробелом.

|  |  |
| --- | --- |
| import sys a = [int(i)for i in input().split()] b = a[-1] #берем послденее значение списка а c = a[0] # берем 0 значение списка а a.append(c) #добавили в конец списка 0-е значение списка a.insert(0,b) #добавили в начало списка -1-е значение списка sum =[] n = len(a)-1 if n ==2 :  print(''.join(str(a[1])))  sys.exit() for i in range(1,n):  sum.append(str(a[i-1]+a[i+1])) print(' '.join(sum)) | a=[int(i) for i in input().split()] n=len(a) if n==1:  print(a[0]) else:  for i in range(n-1):  print(a[i-1]+a[i+1], end=" ")  print(a[n-2]+a[0]) |
|  |  |

Напишите программу, которая принимает на вход список чисел в одной строке и выводит на экран в одну строку значения, которые встречаются в нём более одного раза.

Для решения задачи может пригодиться метод sort списка.

Выводимые числа не должны повторяться, порядок их вывода может быть произвольным.

|  |  |
| --- | --- |
| num\_in = [int(i)for i in input().split()] #вводим числа n\_in\_c = len(num\_in) #подсчитываем колличество строк num\_in.sort() #print(num\_in) k =[] for i in range(0, n\_in\_c) : # `создаем цикл длиной строк   for j in range(0, n\_in\_c):  #print(num\_in[i], j)  if num\_in[i] == num\_in[j] and i != j:  k.append(num\_in[j]) print(\*set(k)) | # a = [1, 1, 3, 3, 2, 3, 3, 3, 3, 4, 4, 6, 7, 7, 9, 9] # r = [] # # for i in range(0, len(a) - 1): # if a[i] == a[i + 1]: # r.append(a[i]) # # s = set(r) # print(\*s) |

Напишите программу, которая считывает с консоли числа (по одному в строке) до тех пор, пока сумма введённых чисел не будет равна 0 и **сразу после этого**выводит сумму квадратов всех считанных чисел.

Гарантируется, что в какой-то момент сумма введённых чисел окажется равной 0, **после этого считывание продолжать не нужно**.

В примере мы считываем числа 1, -3, 5, -6, -10, 13; в этот момент замечаем, что сумма этих чисел равна нулю и выводим сумму их квадратов, не обращая внимания на то, что остались ещё не прочитанные значения.

sum2 = 0  
total =[]  
sum1 = 0  
while True:  
 x = int(input())  
 sum1 += x  
 total.append(x)  
 if sum1 == 0 :  
 break  
for i in total:  
 sum2 += int(pow((i), 2))  
print(sum2)

Напишите программу, которая выводит часть последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ... (число повторяется столько раз, чему равно). На вход программе передаётся неотрицательное целое число n — столько элементов последовательности должна отобразить программа. На выходе ожидается последовательность чисел, записанных через пробел в одну строку.

Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 4.

|  |  |
| --- | --- |
| a = int(input()) b = [] for i in range(0, a + 1):  b += [str(i)] \* i print(\*b[:a]) | a=int(input()) b='' for i in range(0, a + 1):  b += ((str(i) + ' ') \* i) c = b.split() # формирует список чтоб не было проблем с 10… print(c) for i in range(0, a):#перебирает список   print(c[i], end=' ') |

Напишите программу, которая считывает список чисел lst*lst* из первой строки и число x*x* из второй строки, которая выводит все позиции, на которых встречается число x*x* в переданном списке lst*lst*.

Позиции нумеруются с нуля, если число x*x* не встречается в списке, вывести строку "Отсутствует" (без кавычек, с большой буквы).

Позиции должны быть выведены в одну строку, по возрастанию абсолютного значения.

lst = [int(i) for i in input().split()]  
x = int(input())  
out = ''  
for i in range(0, len(lst)):  
 if lst[i] == x:  
 out += str(i) + ' '  
if out == '':  
 print('Отсуствует')  
else:  
 print(, out)

l = [int(i) for i in input().split()]  
x = int(input())  
if not x in l: print('Отсутствует')  
for i in range(len(l)):  
 if l[i]==x: print(i, end = ' ')