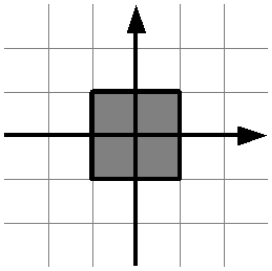
Синтаксис Python. Функции, рекурсия.

Справочный материал можно найти в книжке (ссылка, страница 47.)
Задачи `A-H` требуется оформить следующим образом: написать функцию, название которой указано в условии, принимающую и возвращающую в точности описанные значения.
После описания функции и двух пустых строк написать строчку:
`exec(open('input.txt').read())`

A. *Длина отрезка*
Даны четыре вещественных числа: x_1, y_1, x_2, y_2. Напишите функцию `Distance(x1, y1, x2, y2)`, принимающую на вход четыре вещественных числа и возвращающую расстояние между точкой (x_1, y_1) и (x_2, y_2).

Input	Output
`print(Distance(0.0, 0.0, 1.0, 1.0))`	`1.4142135623730951`

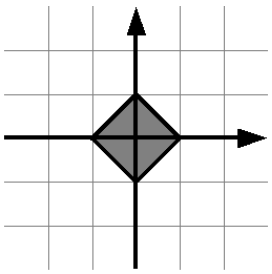
B. *Принадлежит ли точка квадрату - I*
Даны два вещественных числа x и y. Проверьте, принадлежит ли точка с координатами (x, y) заштрихованному квадрату (включая его границу). На рисунке сетка проведена с шагом 1.



Решение должно содержать функцию `IsPointInSquare(x, y)`, принимающую на вход два вещественных числа и возвращающую логическое значение `True`, если точка принадлежит квадрату и логическое значение `False`, если не принадлежит.
Функция `IsPointInSquare` не должна содержать инструкцию `if`.

Input	Output
`print(IsPointInSquare(0.0, 0.0))`	`True`
`print(IsPointInSquare(3.0, -7.0))`	`False`

C. *Принадлежит ли точка квадрату - II*
Решите аналогичную задачу для такого квадрата:

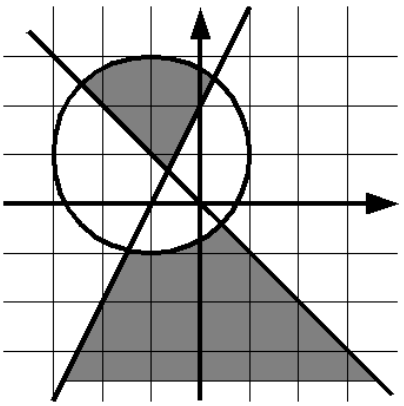


Решение должно содержать функцию `IsPointInSquare(x, y)`, принимающую на вход два вещественных числа и возвращающую логическое значение `True`, если точка принадлежит квадрату и логическое значение `False`, если не принадлежит.
Функция `IsPointInSquare` не должна содержать инструкцию `if`.

Input	Output
`print(IsPointInSquare(0.0, 0.0))`	`True`
`print(IsPointInSquare(1.0, 1.0))`	`False`

D. *Принадлежит ли точка области*

Проверьте, принадлежит ли точка данной закрашенной области:



Решение оформите в виде функции `IsPointInArea(x, y)`.

Решение должно соответствовать требованиям для решения задачи `B`.

Input	Output
`print(IsPointInArea(-4.0, -4.0))`	`False`
`print(IsPointInArea(-1.0, 2.0))`	`True`

E. *Минимальный делитель числа*

Дано натуральное число N ($1 < N < 10^9$). Выведите его наименьший делитель, отличный от 1.

Решение оформите в виде функции `MinDivisor`, принимающей на вход единственное натуральное число и возвращающей его наименьший делитель, отличный от 1. Алгоритм должен иметь сложность $O(\sqrt{N})$.

Указание: Если у числа N не нашлось делителя, ме́ньшего превосходящего $\sqrt{N}$, то число N — простое.

В решении нельзя использовать функцию `sqrt`.

Input	Output
`print(MinDivisor(4))`	`2`
`print(MinDivisor(9409))`	`97`
`print(MinDivisor(59))`	`59`

F. *Проверка числа на простоту*

Дано натуральное число N ($1 < N < 10^9$). Проверьте, является ли оно простым.

Решение оформите в виде функции `IsPrime`, которая принимает на вход единственное число и возвращает логическое значение `True` для простых чисел и логическое значение `False` для составных чисел.

Решение должно иметь сложность $O(\sqrt{N})$.

Input	Output
`print(IsPrime(4))`	`False`
`print(IsPrime(97))`	`True`

G. *Сумма делителей числа*

Для данного натурального числа N ($1 \leqslant N < 10^{10}$) требуется вычислить сумму его делителей, ме́ньших самого числа.

Решение оформите в виде функции `SumDivisors`, принимающей на вход натуральное число и возвращающая сумму его делителей, отличных от него самого.

Решение должно иметь сложность $O(\sqrt{N})$.

Input	Output
`print(SumDivisors(12))`	`16`

H. *Дружественные числа*

Дружественные числа — это два различных натуральных числа, таких, что сумма всех делителей одного числа (меньших самого этого числа) равна другому числу, и наоборот (дружественными являются, например, 220 и 284).

Напишите функцию `IsFriend`, которая проверяет пару чисел на «дружественность» и возвращает логическое значение `True`, если пара чисел дружественная и `False` в противном случае. Функция `IsFriend` должна использовать функцию `SumDivisors` из предыдущей задачи.

Input	Output
`print(IsFriend(220, 284))`	`True`

I. *Дружественные числа в диапазоне*

Дружественные числа — это два натуральных числа, таких, что сумма всех делителей одного числа (меньших самого этого числа) равна другому числу, и наоборот (дружественными являются, например, 220 и 284).

Напишите программу, которая находит все пары не равных друг другу дружественных чисел в заданном диапазоне. Используйте функцию, которая вычисляет сумму делителей числа.

На вход программе подаётся две строки с натуральными числами a и b ($a < b$).

Программа должна вывести пары различных дружественных чисел, каждое из которых находится на отрезке $[a, b]$.

В каждой паре сначала выводится мéньшее число. Пары чисел должны выводиться в порядке возрастания первого числа из пары и разделяться запятой. Каждая пара заключена в скобки.

В случае, если таких пар в указанном диапазоне нет, вывести число 0.

Input	Output
1000 5000	(1184,1210) (2620,2924)

J. *Сумма цифр не меняется*

Напишите программу, которая находит все числа в диапазоне от a до b, сумма цифр которых не меняется при умножении на $2, 3, 4, 5, 6, 7, 8$ и 9 (например, число 9). Используйте функцию для вычисления суммы цифр числа.

На вход программе подаётся две строки с натуральными числами a и b ($a < b$).

Выведите числа, соответствующие условию задачи, в возрастающем порядке через пробел. Если на указанном отрезке таких чисел нет — ничего не выводите.

Input	Output
9 90	9 18 45 90

K. *Гиперпростые числа*

Простое число называется *гиперпростым*, если любое число, получающееся из него откидыванием нескольких последних цифр, тоже является простым. Например, число 733 — гиперпростое, так как и оно само, и числа 73 и 7 — простые. Напишите программу, которая определяет, верно ли, что переданное ей число N — гиперпростое.

Учтите, что число 1 не считается простым.

Input	Output
733	YES

L. *Гиперпростые числа в диапазоне*

Напишите программу, которая находит все гиперпростые числа в заданном диапазоне.

Если в указанном диапазоне гиперпростых чисел нет, вывести число 0.

Input	Output
30 100	31 37 53 59 71 73 79