Вопрос №1: Определение функций на языке Python. Область видимости переменных в пределах модуля, отдельной функции и встроенных (nested) функций.

Синтаксис для определения функций:

def func\_name(args...):

// код. Стоит заметить, что отступ внутри тела функции (как и внутри всякого блока соо своей областью видимости) составляет некоторое количество табов/пробелов, обычно 1 таб / 2 или 4 пробела.

Можно задавать типы аргументов с помощью

def func\_name(arg:type): -> return\_type

Аргументы по умолчанию описываются так:

def func\_name(arg1, default=value):

Как и в С++, аргументы по умолчанию задаются справа налево.

Аргументы также можно задавать как ключевые слова, и они идут после позиционных:

def fn(pos\_args..., \*\*kwargs)

\*\* указывает на упаковку остальных аргументов в словарь. При вызове это выглядит как:

fn(1, 2, position=(0,0), display=True...)

Область видимости переменных

Как правило, в отдельном модуле переменная видна и используется в пределах своего блока. Пр.:

for i in range(0, N):

i видна

i не видна

При вложенных (nested) блоках правило сохраняется. Функция также выступает как отдельный блок. при этом есть нюансы при использовании аргументов. Если аргумент функции - примитив, то внутри блока функции его изменить нельзя, и вне функции значение осталнется тем же. Если же это сложная структура вроде словаря, то все модификации останутся и вне тела функции, так как передача по ссылке позволяет модифицироват исходные данные. Связано это с отсутствием в скриптовых языках полной и правильной реализации глубокого копирования обьектов (сериализация тут не поможет, потому что не сохранятся методы).

Вложенные функции. Имеются в виду лямбда-выражения? Так как лямбда-выражение - это функция, сохраненная в переменную, и ведущая себя соответственно, то можно сделать любую степень вложенности с правилами как для блоков.

square = lambda x: x\*x

Вопрос №2: Напишите программу, которая принимает N-списков и выводит все элементы первого, которых нет во всех других.

test.py