Лабораторная работа №1

Тема: «Работа со встроенными типами данных, числовые типы, строки, кортежи, измененяемые последовательности. Применение основных арифметических операций, определение приоритетов».

Требования к выполнению лабораторной работы №1

1. Изучите теоретическую часть к первой лабораторной работе:
    a. Введение в язык программирования Python (notebook_1.ipynb).
    b. Теоретическая часть к первой лабораторной работе.
    c. Советы по производительности.
    d. Лекция №1.
2. Установите Python + IDE.
3. Создайте новый проект.
4. Запустите примеры из лабораторной работы (notebook_1.ipynb).
5. Выполните задание согласно вашему варианту:
    a. Вычислите свой вариант (*согласно формуле ниже*). Если сделали не свой вариант => работа не засчитывается.
    b. Каждое задание представляет собой отдельный скрипт формата: `lab_{номер_ЛР}_{номер_задания}_{номер_варианта}.py`, пример: `lab_1_2_2.py`
    c. Отправьте выполненное задание в ОРИОКС (*раздел Домашние задания*).

Формат защиты лабораторных работ:

1. Продемонстрируйте выполненные задания.
2. Ответьте на вопросы по вашему коду.
3. При необходимости выполните дополнительное (*дополнительные*) задания от преподавателя.
4. Ответьте (*устно*) преподавателю на контрольные вопросы.

FAQ

- **ЛР №1** можно сдать на максимальный балл на втором занятии при условии, что на первом занятии было выполнено не менее 50% заданий (*оценивается преподавателем*).
- Можно выполнять ЛР опережая график.
- На консультации можно сдавать только долги по ЛР.
- Если вы успели выполнить ЛР (*отправили работу в ОРИОКС*), но по объективным причинам не успели защитить её до конца ЛР, то на следующем занятии баллы не снижаются.
- Баллы не снижаются за отсутствие на занятии по уважительной причине (*в ОРИОКСе должна присутствовать отметка о предоставленной справке*).

Список вопросов

1. Для чего необходимо устанавливать различные версии интерпретатора отдельно для каждого проекта?
2. Что такое инструкция?
3. Что означает термин «динамическая типизация»?
4. В чём отличие *list* от *tuple*?
5. В чём отличие *set* от *list*?
6. Что возвращает функция *range*?

Задания

Во всех заданиях необходимо проверять корректность вводимых данных и выводить соответствующие сообщения об ошибках.

№ Варианта = *номер_студенческого* % 2 + 1

Вариант №1

Задание №1. Пользователь с клавиатуры вводит две строки *s* и *x*, где *s* – исходная строка, *x* – «вирус». Необходимо из исходной строки s удалить все вхождения строки *x*, таким образом, чтобы в результирующей строке не осталось «вирусов».

** Строки не чувствительны к регистру.*

Примеры:

s	*x*	*result*
python	py	thon
tuple	Up	tle
Queues	ue	Qs
aabbcc	ab	cc

Задание №2. Пользователь с клавиатуры вводит *n* целых положительных чисел (*через пробел*) – получаем список a_n.

Необходимо, чтобы после *k* операций все элементы списка равнялись нулю.

Ограничение: обнулять элементы можно только с помощью следующей операции: скрипт автоматически выбирает оптимальный отрезок $[a_i, a_j]$, такой, что $1 \leqslant i \leqslant j \leqslant n$, затем уменьшает элементы $a_i, a_{i+1}, a_{i+2}, \ldots, a_j$ на единицу.

Задача вывести число *k* – минимальное количество операций, после которых все элементы списка будут равны нулю.

Примеры:

a	*k*	*Примечание*
[2, 2]	2	[2, 2] => [1, 1] => [0, 0]
[4, 4, 5, 5]	5	[4, 4, 5, 5] => [3, 3, 4, 4] => [2, 2, 3, 3] => [1, 1, 2, 2] => [0, 0, 1, 1] => [0, 0, 0, 0]
[4, 2, 4]	6	[4, 2, 4] => [3, 1, 3] => [2, 0, 2] => [1, 0, 2] => [0, 0, 2] => [0, 0, 1] => [0, 0, 0]

Задание №3.

Администратору кинотеатра необходимо вести учёт стоимости билетов. Цена на билет в различные дни/часы может изменяться, поэтому для пары ряд/место, может быть задано несколько строк.

Входные данные:

Пользователь вводит целое положительное число *n* – количество строк.

Затем вводит n строк формата:

{ряд} {место} {стоимость_билета}

Например, строка

1 2 1000

означает: 1-й ряд, 2 место, 1000 руб.

Выходные данные:

m пар *{ряд$_i$} {место$_i$}* – {k_i}

где k_i – количество различных возможных цен билета на *{ряд$_i$} {место$_i$}*

Примеры:

n	*tickets*	*k*	*Примечание*
4	1 1 1000 1 1 1000 1 2 2000 1 2 3000	1 1 – 1 1 2 – 2	Билет на 1-й ряд 1 место во всех кейсах имеет только одну цену (1000 руб.) => $k_1 = 1$ Билет на 1-й ряд 2 место имеет 2 различные цены (2000 руб. и 3000 руб.) => $k_2 = 2$

3	1 1 1000 1 1 2000 1 1 2000	1 1 – 2	Билет на 1-й ряд 1 место имеет две различные цены (1000 руб. и 2000 руб.) => $k_1 = 2$

Примечание: в качестве ключей словаря (*dict*) могут быть использованы кортежи (*tuple*).

Вариант №2

З**адание №1.** Пользователь с клавиатуры вводит строку *s (разрешаются только латинские символы без пробелов)*. Необходимо вывести целое число *n* – количество индексов *i* таких, что после удаления символа s_i из исходной строки *s* новая строка *s'* становится палиндромом. Палиндромом называется строка, которая одинаково читается как слева направо, так и справа налево.

Примеры:

s	*n*	*Примечание*
dad	1	При удалении первого символа: ad – не палиндром При удалении второго символа: dd – **палиндром** При удалении третьего символа: da – не палиндром
qq	2	При удалении первого символа: d – **палиндром** При удалении второго символа: d – **палиндром**
Level	1	evel – не палиндром Lvel – не палиндром Leel – **палиндром** Levl – не палиндром Leve – не палиндром

Задание №2. Пользователь с клавиатуры вводит целые неотрицательные числа (*через пробел*). Необходимо отсортировать список чисел по количеству вхождений единиц в бинарном представлении числа. Если два числа имеют одинаковое количество единиц (*битов*), вместо этого сравните их реальные значения.

Примеры:

list	*result*	*Примечание*
[2, 1, 3]	[1, 2, 3]	1 – 0**1**; 2 – **1**0; 3 – **11** 1 и 2 имеют одинаковое количество «единиц», но так как 2 > 1 => число 1 имеет меньший приоритет.
[8, 16]	[8, 16]	8 – **1**000; 16 – **1**0000
[15, 32, 63, 64]	[32, 64, 15, 63]	32 – 0**1**00000; 64 – **1**000000; 15 – 000**1111**; 63 – 0**111111**;

Задание №3.

Сетевому администратору необходимо проанализировать логи авторизации. Каждая строка такого лога представляет себе данные о дате авторизации пользователя и его IP адресе.

Входные данные:

Пользователь вводит целое положительное число *n* – количество строк.

Затем вводит n строк формата:

{логин} {дата} {IP}

Login – набор латинский букв без пробелов (например, Admin).

Дата – дата формата dd.mm.YYYY (например, 31.12.2021).

IP – IP адрес формата IPv4 (например, 10.0.0.2).

Выходные данные:

Вывести *логин* с максимальным количеством различных IP адресов за один день. Если таких несколько, то вывести любой из них.

Примеры:

n	*Логи*	*Логин*	*Примечание*
5	Admin 01.01.2021 10.0.0.2 Admin 01.01.2021 10.0.0.3 User 01.01.2021 192.168.0.1 User 02.01.2021 192.168.0.2 User 03.01.2021 192.168.0.3	Admin	01.01.2021 пользователь с логином Admin авторизовался с двух различных адресов. Пользователь с логином User хоть и авторизовался с 3-х различных адресов, но все в разные дни.
4	Admin 01.01.2021 10.0.0.2 User 02.01.2021 192.168.0.1 User 02.01.2021 192.168.0.2 User 02.01.2021 192.168.0.3	User	Пользователь User авторизовался с 3-х различных IP адресов за один день (02.01.2021)

Примечание: в качестве ключей словаря (*dict*) могут быть использованы кортежи (*tuple*). Проверять корректность IP адреса и/или даты желательно, но не является обязательной частью задания.