№1

Easy

# Задача-1: поработайте с переменными, создайте несколько,

# выведите на экран, запросите от пользователя и сохраните в переменную, выведите на экран

# Задача-2: Запросите от пользователя число, сохраните в переменную,

# прибавьте к числу 2 и выведите результат на экран.

# Если возникла ошибка, прочитайте ее, вспомните урок и постарайтесь устранить ошибку.

# Задача-3: Запросите у пользователя его возраст.

# Если ему есть 18 лет, выведите: "Доступ разрешен",

# иначе "Извините, пользование данным ресурсом только с 18 лет"

Normal

# Задача: используя цикл запрашивайте у пользователя число пока оно не станет больше 0, но меньше 10.

# После того, как пользователь введет корректное число, возведите его в степерь 2 и выведите на экран.

# Например, пользователь вводит число 123, вы сообщаете ему, что число не верное,

# и сообщаете об диапазоне допустимых. И просите ввести заного.

# Допустим пользователь ввел 2, оно подходит, возводим в степень 2, и выводим 4

# Задача-2: Исходные значения двух переменных запросить у пользователя.

# Поменять значения переменных местами. Вывести новые значения на экран.

# Решите задачу, используя только две переменные.

# Подсказки:

# \* постарайтесь сделать решение через действия над числами;

Hard

# Создайте программу медицинская анкета, где вы запросите у пользователя такие данные, как имя, фамилию, возраст, и вес.

# И выведите результат согласно которому пациент в хорошем состоянии, если ему до 30 лет и вес от 50 и до 120 кг,

# Пациенту требуется начать вести правильный образ жизни, если ему более 30 и вес меньше 50 или больше 120 кг

# Пациенту требуется врачебный осмотр, если ему более 40 и вес менее 50 или больше 120 кг.

# Все остальные варианты вы можете обработать на ваш вкус и полет фантазии =)

# Формула не отражает реальной действительности и здесь используется только ради примера.

# Пример: Вася Пупкин, 29 год, вес 90 - хорошее состояние

# Пример: Вася Пупкин, 31 год, вес 121 - следует заняться собой

# Пример: Вася Пупкин, 31 год, вес 49 - следует заняться собой

# Пример: Вася Пупкин, 41 год, вес 121 - следует обратится к врачу!

# Пример: Вася Пупкин, 41 год, вес 49 - следует обратится к врачу!

#2

Easy

|  |
| --- |
| # Задача-1: |
|  |

|  |
| --- |
| # Дан список фруктов. |
|  |

|  |
| --- |
| # Напишите программу, выводящую фрукты в виде нумерованного списка, |
|  |

|  |
| --- |
| # выровненного по правой стороне. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Пример: |
|  |

|  |
| --- |
| # Дано: ["яблоко", "банан", "киви", "арбуз"] |
|  |

|  |
| --- |
| # Вывод: |
|  |

|  |
| --- |
| # 1. яблоко |
|  |

|  |
| --- |
| # 2. банан |
|  |

|  |
| --- |
| # 3. киви |
|  |

|  |
| --- |
| # 4. арбуз |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Подсказка: воспользоваться методом .format() |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-2: |
|  |

|  |
| --- |
| # Даны два произвольные списка. |
|  |

|  |
| --- |
| # Удалите из первого списка элементы, присутствующие во втором списке. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-3: |
|  |

|  |
| --- |
| # Дан произвольный список из целых чисел. |
|  |

|  |
| --- |
| # Получите НОВЫЙ список из элементов исходного, выполнив следующие условия: |
|  |

# если элемент кратен двум, то разделить его на 4, если не кратен, то умножить на два.

Normal

|  |
| --- |
| # Задача-1: |
|  |

|  |
| --- |
| # Дан список, заполненный произвольными целыми числами, получите новый список, |
|  |

|  |
| --- |
| # элементами которого будут квадратные корни элементов исходного списка, |
|  |

|  |
| --- |
| # но только если результаты извлечения корня не имеют десятичной части и |
|  |

|  |
| --- |
| # если такой корень вообще можно извлечь |
|  |

|  |
| --- |
| # Пример: Дано: [2, -5, 8, 9, -25, 25, 4] Результат: [3, 5, 2] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-2: Дана дата в формате dd.mm.yyyy, например: 02.11.2013. |
|  |

|  |
| --- |
| # Ваша задача вывести дату в текстовом виде, например: второе ноября 2013 года. |
|  |

|  |
| --- |
| # Склонением пренебречь (2000 года, 2010 года) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-3: Напишите алгоритм, заполняющий список произвольными целыми числами |
|  |

|  |
| --- |
| # в диапазоне от -100 до 100. В списке должно быть n - элементов. |
|  |

|  |
| --- |
| # Подсказка: |
|  |

|  |
| --- |
| # для получения случайного числа используйте функцию randint() модуля random |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-4: Дан список, заполненный произвольными целыми числами. |
|  |

|  |
| --- |
| # Получите новый список, элементами которого будут: |
|  |

|  |
| --- |
| # а) неповторяющиеся элементы исходного списка: |
|  |

|  |
| --- |
| # например, lst = [1, 2, 4, 5, 6, 2, 5, 2], нужно получить lst2 = [1, 2, 4, 5, 6] |
|  |

|  |
| --- |
| # б) элементы исходного списка, которые не имеют повторений: |
|  |

# например, lst = [1 , 2, 4, 5, 6, 2, 5, 2], нужно получить lst2 = [1, 4, 6]

Hard

|  |
| --- |
| # Задание-1: уравнение прямой вида y = kx + b задано в виде строки. |
|  |

|  |
| --- |
| # Определить координату y точки с заданной координатой x. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| equation = 'y = -12x + 11111140.2121' |
|  |

|  |
| --- |
| x = 2.5 |
|  |

|  |
| --- |
| # вычислите и выведите y |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задание-2: Дата задана в виде строки формата 'dd.mm.yyyy'. |
|  |

|  |
| --- |
| # Проверить, корректно ли введена дата. |
|  |

|  |
| --- |
| # Условия корректности: |
|  |

|  |
| --- |
| # 1. День должен приводиться к целому числу в диапазоне от 1 до 30(31) |
|  |

|  |
| --- |
| # (в зависимости от месяца, февраль не учитываем) |
|  |

|  |
| --- |
| # 2. Месяц должен приводиться к целому числу в диапазоне от 1 до 12 |
|  |

|  |
| --- |
| # 3. Год должен приводиться к целому положительному числу в диапазоне от 1 до 9999 |
|  |

|  |
| --- |
| # 4. Длина исходной строки для частей должна быть в соответствии с форматом |
|  |

|  |
| --- |
| # (т.е. 2 символа для дня, 2 - для месяца, 4 - для года) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Пример корректной даты |
|  |

|  |
| --- |
| date = '01.11.1985' |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Примеры некорректных дат |
|  |

|  |
| --- |
| date = '01.22.1001' |
|  |

|  |
| --- |
| date = '1.12.1001' |
|  |

|  |
| --- |
| date = '-2.10.3001' |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задание-3: "Перевёрнутая башня" (Задача олимпиадного уровня) |
|  |

|  |
| --- |
| # |
|  |

|  |
| --- |
| # Вавилонцы решили построить удивительную башню — |
|  |

|  |
| --- |
| # расширяющуюся к верху и содержащую бесконечное число этажей и комнат. |
|  |

|  |
| --- |
| # Она устроена следующим образом — на первом этаже одна комната, |
|  |

|  |
| --- |
| # затем идет два этажа, на каждом из которых по две комнаты, |
|  |

|  |
| --- |
| # затем идёт три этажа, на каждом из которых по три комнаты и так далее: |
|  |

|  |
| --- |
| # ... |
|  |

|  |
| --- |
| # 12 13 14 |
|  |

|  |
| --- |
| # 9 10 11 |
|  |

|  |
| --- |
| # 6 7 8 |
|  |

|  |
| --- |
| # 4 5 |
|  |

|  |
| --- |
| # 2 3 |
|  |

|  |
| --- |
| # 1 |
|  |

|  |
| --- |
| # |
|  |

|  |
| --- |
| # Эту башню решили оборудовать лифтом --- и вот задача: |
|  |

|  |
| --- |
| # нужно научиться по номеру комнаты определять, |
|  |

|  |
| --- |
| # на каком этаже она находится и какая она по счету слева на этом этаже. |
|  |

|  |
| --- |
| # |
|  |

|  |
| --- |
| # Входные данные: В первой строчке задан номер комнаты N, 1 ≤ N ≤ 2 000 000 000. |
|  |

|  |
| --- |
| # |
|  |

|  |
| --- |
| # Выходные данные: Два целых числа — номер этажа и порядковый номер слева на этаже. |
|  |

|  |
| --- |
| # |
|  |

|  |
| --- |
| # Пример: |
|  |

|  |
| --- |
| # Вход: 13 |
|  |

|  |
| --- |
| # Выход: 6 2 |
|  |

|  |
| --- |
| # |
|  |

|  |
| --- |
| # Вход: 11 |
|  |

# Выход: 5 3

#3

Easy

# Постарайтесь использовать то, что мы прошли на уроке при решении этого ДЗ,

# вспомните про zip(), map(), lambda, посмотрите где лучше с ними, а где они излишни!

# Задание - 1

# Создайте функцию, принимающую на вход Имя, возраст и город проживания человека

# Функция должна возвращать строку вида "Василий, 21 год(а), проживает в городе Москва"

# Задание - 2

# Создайте функцию, принимающую на вход 3 числа, и возвращающую наибольшее из них

# Задание - 3

# Создайте функцию, принимающую неограниченное количество строковых аргументов,

# верните самую длинную строку из полученных аргументов

Normal

# Задание - 1

# Вам даны 2 списка одинаковой длины, в первом списке имена людей, во втором зарплаты,

# вам необходимо получить на выходе словарь, где ключ - имя, значение - зарплата.

# Запишите результаты в файл salary.txt так, чтобы на каждой строке было 2 столбца,

# столбцы разделяются пробелом, тире, пробелом. в первом имя, во втором зарплата, например: Vasya - 5000

# После чего прочитайте файл, выведите построчно имя и зарплату минус 13% (налоги ведь),

# Есть условие, не отображать людей получающих более зарплату 500000, как именно

#  выполнить условие решать вам, можете не писать в файл

# можете не выводить, подумайте какой способ будет наиболее правильным и оптимальным,

#  если скажем эти файлы потом придется передавать.

# Так же при выводе имя должно быть полностью в верхнем регистре!

# Подумайте вспоминая урок, как это можно сделать максимально кратко, используя возможности языка Python.

Hard

# Задание - 1

# Давайте опишем пару сущностей player и enemy через словарь,

# который будет иметь ключи и значения:

# name - строка полученная от пользователя,

# health - 100,

# damage - 50.

# Поэксперементируйте с значениями урона и жизней по желанию.

# Теперь надо создать функцию attack(person1, person2), аргументы можете указать свои,

# функция в качестве аргумента будет принимать атакующего и атакуемого,

# функция должна получить параметр damage атакующего и отнять это количество

# health от атакуемого. Функция должна сама работать с словарями и изменять их значения.

# Задание - 2

# Давайте усложним предыдущее задание, измените сущности, добавив новый параметр - armor = 1.2

# Теперь надо добавить функцию, которая будет вычислять и возвращать полученный урон по формуле damage / armor

# Следовательно у вас должно быть 2 функции, одна наносит урон, вторая вычисляет урон по отношению к броне.

# Сохраните эти сущности, полностью, каждую в свой файл,

# в качестве названия для файла использовать name, расширение .txt

# Напишите функцию, которая будет считывать файл игрока и его врага, получать оттуда данные, и записывать их в словари,

# после чего происходит запуск игровой сессии, где сущностям поочередно наносится урон,

# пока у одного из них health не станет меньше или равен 0.

# После чего на экран должно быть выведено имя победителя, и количество оставшихся единиц здоровья.

#4 (полезный сайт для проверки регэкспов https://regex101.com/)

Easy

# Все задачи текущего блока решите с помощью генераторов списков!

# Задание-1:

# Дан список, заполненный произвольными целыми числами.

# Получить новый список, элементы которого будут

# квадратами элементов исходного списка

# [1, 2, 4, 0] --> [1, 4, 16, 0]

# Задание-2:

# Даны два списка фруктов.

# Получить список фруктов, присутствующих в обоих исходных списках.

# Задание-3:

# Дан список, заполненный произвольными числами.

# Получить список из элементов исходного, удовлетворяющих следующим условиям:

# + Элемент кратен 3

# + Элемент положительный

# + Элемент не кратен 4

Normal

# Эти задачи необходимо решить используя регулярные выражения!

# Задача - 1

# Запросите у пользователя имя, фамилию, email. Теперь необходимо совершить проверки, имя и фамилия должны иметь заглавные первые буквы.

# email - не должен иметь заглавных букв и должен быть в формате: текст в нижнем регистре, допускается нижнее подчеркивание и цифры, потом @, потом текст, допускаются цифры, точка, ru или org или com.

# Например:

# Пупкин василий - неверно указано имя, te$T@test.net - неверно указан email (спецсимвол, заглавная буква, .net), te\_4\_st@test.com - верно указан.

# Задача - 2:

# Вам дан текст:

some\_str = '''

Мороз и солнце; день чудесный!

Еще ты дремлешь, друг прелестный —

Пора, красавица, проснись:

Открой сомкнуты негой взоры

Навстречу северной Авроры,

Звездою севера явись!

Вечор, ты помнишь, вьюга злилась,

На мутном небе мгла носилась;

Луна, как бледное пятно,

Сквозь тучи мрачные желтела,

И ты печальная сидела —

А нынче... погляди в окно:

Под голубыми небесами

Великолепными коврами,

Блестя на солнце, снег лежит;

Прозрачный лес один чернеет,

И ель сквозь иней зеленеет,

И речка подо льдом блестит.

Вся комната янтарным блеском

Озарена. Веселым треском

Трещит затопленная печь.

Приятно думать у лежанки.

Но знаешь: не велеть ли в санки

Кобылку бурую запречь?

Скользя по утреннему снегу,

Друг милый, предадимся бегу

Нетерпеливого коня

И навестим поля пустые,

Леса, недавно столь густые,

И берег, милый для меня.'''

# Необходимо с помощью регулярных выражений определить есть ли в тексте подряд

# более одной точки, при любом исходе сообщите результат пользователю!

Hard

# Задание:

# Эта программа являет собой упрощенный симулятор банкомата, пользователь вводит номер карты и пин код,

# в случае успеха программа предлагает меню для выбора действий, где он может проверить счет, или снять деньги.

#

# Эта задача не так похожа на другие, но она, как никогда прежде приближена к реалиям разработки общего проекта.

#

# Ваша задача исправить ошибки логики, и выполнить проверки данных, которые вводит пользователь.

# Обязательно убедитесь, что вы выполнили все проверки, попробуйте сами сломать свою программу вводя неверные данные!

person1 = {'card': 4276123465440000, 'pin': 9090, 'money': 100.90}

person2 = {'card': 4276123465440001, 'pin': 9091, 'money': 200.90}

person3 = {'card': 4276123465440002, 'pin': 9092, 'money': 300.90}

bank = [person1, person2, person3]

def get\_person\_by\_card(card\_number):

    for person in bank:

        if person['card'] == card\_number:

            return person

def is\_pin\_valid(person, pin\_code):

    if person['pin'] == pin\_code:

        return True

    return False

def check\_account(person):

    return round(person['money'], 2)

def withdraw\_money(person, money):

    if person['money'] - money == 0:

        person['money'] -= money

        return 'Вы сняли {} рублей.'.format(money)

    else:

        return 'На вашем счету недостаточно средств!'

def process\_user\_choice(choice, person):

    if choice == '1':

        print(check\_account(person))

    elif choice == '2':

        count = float(input('Сумма к снятию:'))

        print(withdraw\_money(person, count))

def start():

    card\_number, pin\_code = input('Введите номер карты и пин код через пробел:').split()

    card\_number = int(card\_number)

    pin\_code = int(pin\_code)

    person = get\_person\_by\_card(card\_number)

    if person and is\_pin\_valid(person, pin\_code):

        while True:

            choice = int(input('Выберите пункт:\n'

                               '1. Проверить баланс\n'

                               '2. Снять деньги\n'

                               '3. Выход\n'

                               '---------------------\n'

                               'Ваш выбор:'))

            if choice == 3:

                break

            process\_user\_choice(choice, person)

    else:

        print('Номер карты или пин код введены не верно!')

start()

#5

Easy

|  |
| --- |
| # Задача-1: |
|  |

|  |
| --- |
| # Напишите скрипт, создающий директории dir\_1 - dir\_9 в папке, |
|  |

|  |
| --- |
| # из которой запущен данный скрипт. |
|  |

|  |
| --- |
| # И второй скрипт, удаляющий эти папки. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-2: |
|  |

|  |
| --- |
| # Напишите скрипт, отображающий папки текущей директории. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Задача-3: |
|  |

# Напишите скрипт, создающий копию файла, из которого запущен данный скрипт.

Normal

|  |
| --- |
| # Задача-1: |
|  |

|  |
| --- |
| # Напишите небольшую консольную утилиту, |
|  |

|  |
| --- |
| # позволяющую работать с папками текущей директории. |
|  |

|  |
| --- |
| # Утилита должна иметь меню выбора действия, в котором будут пункты: |
|  |

|  |
| --- |
| # 1. Перейти в папку |
|  |

|  |
| --- |
| # 2. Просмотреть содержимое текущей папки |
|  |

|  |
| --- |
| # 3. Удалить папку |
|  |

|  |
| --- |
| # 4. Создать папку |
|  |

|  |
| --- |
| # При выборе пунктов 1, 3, 4 программа запрашивает название папки |
|  |

|  |
| --- |
| # и выводит результат действия: "Успешно создано/удалено/перешел", |
|  |

|  |
| --- |
| # "Невозможно создать/удалить/перейти" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Для решения данной задачи используйте алгоритмы из задания easy, |
|  |

|  |
| --- |
| # оформленные в виде соответствующих функций, |
|  |

# и импортированные в данный файл из easy.py

Hard

|  |
| --- |
| # Задание-1: |
|  |

|  |
| --- |
| # Доработайте реализацию программы из примера examples/5\_with\_args.py, |
|  |

|  |
| --- |
| # добавив реализацию следующих команд (переданных в качестве аргументов): |
|  |

|  |
| --- |
| # cp <file\_name> - создает копию указанного файла |
|  |

|  |
| --- |
| # rm <file\_name> - удаляет указанный файл (запросить подтверждение операции) |
|  |

|  |
| --- |
| # cd <full\_path or relative\_path> - меняет текущую директорию на указанную |
|  |

|  |
| --- |
| # ls - отображение полного пути текущей директории |
|  |

|  |
| --- |
| # путь считать абсолютным (full\_path) - |
|  |

|  |
| --- |
| # в Linux начинается с /, в Windows с имени диска, |
|  |

|  |
| --- |
| # все остальные пути считать относительными. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Важно! Все операции должны выполняться в той директории, в который вы находитесь. |
|  |

|  |
| --- |
| # Исходной директорией считать ту, в которой был запущен скрипт. |
|  |

|  |
| --- |
|  |
|  |

# P.S. По возможности, сделайте кросс-платформенную реализацию.

#6

Easy

# Задача - 1

# Опишите несколько классов TownCar, SportCar, WorkCar, PoliceCar

# У каждого класса должны быть следующие аттрибуты:

# speed, color, name, is\_police - Булево значение.

# А так же несколько методов: go, stop, turn(direction) - которые должны сообщать,

#  о том что машина поехала, остановилась, повернула(куда)

# Задача - 2

# Посмотрите на задачу-1 подумайте как выделить общие признаки классов

# в родительский и остальные просто наследовать от него.

Normal

# Задача - 1

# Ранее мы с вами уже писали игру, используя словари в качестве

# структур данных для нашего игрока и врага, давайте сделаем новую, но уже с ООП

# Опишите базовый класс Person, подумайте какие общие данные есть и у врага и у игрока

# Не забудьте, что у них есть помимо общих аттрибутов и общие методы.

# Теперь наследуясь от Person создайте 2 класса Player, Enemy.

# У каждой сущности должы быть аттрибуты health, damage, armor

# У каждой сущности должно быть 2 метода, один для подсчета урона, с учетом брони противника,

# второй для атаки противника.

# Функция подсчета урона должна быть инкапсулирована

# Вам надо описать игровой цикл так же через класс.

# Создайте экземпляры классов, проведите бой. Кто будет атаковать первым оставляю на ваше усмотрение.

Hard

# Задача - 1

# Вам необходимо создать завод по производству мягких игрушек для детей.

# Вам надо продумать структуру классов,

# чтобы у вас был класс, который создает игрушки на основании:

#  Названия, Цвета, Типа (животное, персонаж мультфильма)

# Опишите процедуры создания игрушки в трех методах:

# -- Закупка сырья, пошив, окраска

# Не усложняйте пусть методы просто выводят текст о том, что делают.

# В итоге ваш класс по производству игрушек должен вернуть объект нового класса Игрушка.

# Задача - 2

# Доработайте нашу фабрику, создайте по одному классу на каждый тип, теперь надо в классе фабрика

# исходя из типа игрушки отдавать конкретный объект класса, который наследуется от базового – Игрушка

#7

|  |
| --- |
| """ |
|  |

|  |
| --- |
| == Лото == |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Правила игры в лото. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Игра ведется с помощью специальных карточек, на которых отмечены числа, |
|  |

|  |
| --- |
| и фишек (бочонков) с цифрами. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Количество бочонков — 90 штук (с цифрами от 1 до 90). |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Каждая карточка содержит 3 строки по 9 клеток. В каждой строке по 5 случайных цифр, |
|  |

|  |
| --- |
| расположенных по возрастанию. Все цифры в карточке уникальны. Пример карточки: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| -------------------------- |
|  |

|  |
| --- |
| 9 43 62 74 90 |
|  |

|  |
| --- |
| 2 27 75 78 82 |
|  |

|  |
| --- |
| 41 56 63 76 86 |
|  |

|  |
| --- |
| -------------------------- |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| В игре 2 игрока: пользователь и компьютер. Каждому в начале выдается |
|  |

|  |
| --- |
| случайная карточка. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Каждый ход выбирается один случайный бочонок и выводится на экран. |
|  |

|  |
| --- |
| Также выводятся карточка игрока и карточка компьютера. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Пользователю предлагается зачеркнуть цифру на карточке или продолжить. |
|  |

|  |
| --- |
| Если игрок выбрал "зачеркнуть": |
|  |

|  |
| --- |
| Если цифра есть на карточке - она зачеркивается и игра продолжается. |
|  |

|  |
| --- |
| Если цифры на карточке нет - игрок проигрывает и игра завершается. |
|  |

|  |
| --- |
| Если игрок выбрал "продолжить": |
|  |

|  |
| --- |
| Если цифра есть на карточке - игрок проигрывает и игра завершается. |
|  |

|  |
| --- |
| Если цифры на карточке нет - игра продолжается. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Побеждает тот, кто первый закроет все числа на своей карточке. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Пример одного хода: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Новый бочонок: 70 (осталось 76) |
|  |

|  |
| --- |
| ------ Ваша карточка ----- |
|  |

|  |
| --- |
| 6 7 49 57 58 |
|  |

|  |
| --- |
| 14 26 - 78 85 |
|  |

|  |
| --- |
| 23 33 38 48 71 |
|  |

|  |
| --- |
| -------------------------- |
|  |

|  |
| --- |
| -- Карточка компьютера --- |
|  |

|  |
| --- |
| 7 11 - 14 87 |
|  |

|  |
| --- |
| 16 49 55 77 88 |
|  |

|  |
| --- |
| 15 20 - 76 - |
|  |

|  |
| --- |
| -------------------------- |
|  |

|  |
| --- |
| Зачеркнуть цифру? (y/n) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Подсказка: каждый следующий случайный бочонок из мешка удобно получать |
|  |

|  |
| --- |
| с помощью функции-генератора. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Подсказка: для работы с псевдослучайными числами удобно использовать |
|  |

|  |
| --- |
| модуль random: http://docs.python.org/3/library/random.html |
|  |

|  |
| --- |
|  |
|  |

"""