Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №1**

по дисциплине: «Теория алгоритмов»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Мельников М.Н. |
| Проверила:  старший преподаватель кафедры ПО  Корнеева Е.И. |

Тверь 2024

# Задачи.

## Расчет расстояний между городами.

### Описание задачи:

Необходимо вычислить расстояния между несколькими городами, используя их координаты. В результате требуется создать словарь, где ключом будет название города, а значением — другой словарь с расстояниями до всех остальных городов. Формула для расчета расстояния между двумя городами выглядит так: (x1−x2)2+(y1−y2)2\sqrt{(x1 - x2)^2 + (y1 - y2)^2}(x1−x2)2+(y1−y2)2​.

### Описание выполненной работы.

В этой задаче мы сделали программу, которая считает расстояния между городами по их координатам. Сначала пользователю нужно ввести, сколько городов он хочет добавить и их названия с координатами. Затем программа считает расстояния и выводит все это в удобном виде.

Функция calcDistance(cord1, cord2):

Возвратить квадратный корень из ((cord1[0] - cord2[0]) в квадрате + (cord1[1] - cord2[1]) в квадрате)

Функция inputLocation():

sites ← Пустой словарь

Вывести "Кол-во городов: "

sitesCount ← Число, введенное пользователем

Для i от 0 до sitesCount - 1:

Вывести "Название города и координаты. Формат - CityName x:y "

inpSites ← Строка, введенная пользователем

cityName ← Первый элемент inpSites, разделенный по пробелу

coordinates ← Второй элемент inpSites, разделенный по пробелу

x ← Первый элемент coordinates, разделенный по ":"

y ← Второй элемент coordinates, разделенный по ":"

sites[cityName] ← (Целое значение x, Целое значение y)

Вывести "Город " + cityName + " добавлен!"

Возвратить sites

Функция fillDistance(sites):

distance ← Пустой словарь

Для каждого city1, coord1 в sites:

distance[city1] ← Пустой словарь

Для каждого city2, coord2 в sites:

Если city1 ≠ city2:

distance[city1][city2] ← Округлить(calcDistance(coord1, coord2), 2)

Возвратить distance

Функция showDistance(distance):

Вывести "Расстояния между городами:"

Для каждого city, dests в distance:

Для каждого dest, dist в dests:

Вывести city + " -> " + dest + ": " + dist + " км"

Функция main():

sites ← inputLocation()

distance ← fillDistance(sites)

showDistance(distance)

Если имя программы = "main":

main()

#Расчет расстояния на координатной сетке

def calcDistance(cord1, cord2):

    return math.sqrt(math.pow((cord1[0] - cord2[0]), 2) + math.pow((cord1[1] - cord2[1]), 2))

#Ввод расположения городов на координатной сетке

def inputLocation():

    sites = {}

    sitesCount = int(input("Кол-во городов: "))

    for i in range(sitesCount):

        inpSites = input("Название города и координаты. Формат - CityName x:y ")

        sites[inpSites.split(' ')[0]] = (int(inpSites.split(' ')[1].split(':')[0]), int(inpSites.split(' ')[1].split(':')[1]))

        print(f"Город {inpSites.split(' ')[0]} добавлен!")

    return sites

#Заполнение словаря расстояний между городами

def fillDistance(sites):

    distance = {}

    for city1, coord1 in sites.items():

        distance[city1] = {}

        for city2, coord2 in sites.items():

            if city1 != city2:

                distance[city1][city2] = round(calcDistance(coord1, coord2), 2)

    return distance

#Вывод расстояний

def showDistance(distance):

    print("Расстояния между городами:")

    for city, dests in distance.items():

        for dest, distance in dests.items():

            print(f"{city} -> {dest}: {distance} км")

def main():

    showDistance(fillDistance(inputLocation()))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

## Вторая задача

### Постановка

Необходимо вычислить площадь круга с заданным радиусом и проверить, находятся ли определенные точки внутри этого круга. Круг центрирован в начале координат.

### Описание проделанной работы

**Ввод радиуса**: Пользователь вводит радиус круга.

**Ввод точек**: Пользователь вводит координаты двух точек.

**Расчет площади**: Программа считает площадь круга с помощью формулы πr2\pi r^2πr2.

**Проверка точек**: Для каждой точки программа проверяет, находится ли она внутри круга, сравнивая расстояние от точки до центра с радиусом круга.

**Вывод результатов**: Площадь круга и информация о том, находятся ли точки внутри круга, выводятся на экран.

Функция calcSquare(radius):

"""Расчет площади круга"""

Возвратить округленное значение (число Пи \* (radius в квадрате), до 4 знаков после запятой)

Функция calcDistanceToPoint(point):

"""Расчет расстояния от точки до начала координат"""

Возвратить квадратный корень из ((point[0] в квадрате) + (point[1] в квадрате))

Функция isPointInside(point, radius):

"""Проверка нахождения точки в области круга"""

Возвратить результат сравнения (calcDistanceToPoint(point) <= radius)

Функция getRadius():

"""Получение радиуса круга от пользователя"""

Вывести "Введите радиус круга:"

Возвратить число, введенное пользователем

Функция getPoint():

"""Получение расположения точки"""

Вывести "Введите координаты точки (Формат: x-y):"

Возвратить строку, введенную пользователем, разделенную по символу '-'

Функция main():

radius ← getRadius()

coordPoint ← getPoint()

Вывести "Площадь круга: " + calcSquare(radius)

Вывести "Нахождение точки в области круга радиусом " + radius + ": " + isPointInside(coordPoint, radius)

Если имя программы = "main":

main()

import math

def calcSquare(radius):

    """Расчет площади круга"""

    return round(math.pi \* math.pow(radius, 2), 4)

def calcDistanceToPoint(point):

    """Расчет расстояния от точки до начала координат"""

    return math.sqrt(float(point[0]) \*\* 2 + float(point[1]) \*\* 2)

def isPointIndside(point, radius):

    """Проверка нахождения точки в области круга"""

    return calcDistanceToPoint(point) <= radius

def getRadius():

    """Получение радиуса круга от пользователя"""

    return float(input("Введите радиус круга:"))

def getPoint():

    """Получение располочение точки"""

    return (input("Введите координаты точки (Формат: x-y)")).split('-')

def main():

    radius = getRadius()

    coordPoint = getPoint()

    print(f"Площадь круга: {calcSquare(radius)}")

    print(f"Нахождение в точки в области круга радиусом {radius} - {isPointIndside(coordPoint, radius)}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

## Третья задача

### Постановка

Необходимо расставить знаки операций "+", "-", "\*" и скобки между числами "1 2 3 4 5", чтобы в итоге получилось число "25". Порядок чисел при этом должен сохраняться.

### Описание проделанной работы

Удалось найти подходящее выражение, используя предложенные операции и сохранив порядок чисел.

result = (1\*(2+(3+(4\*5)))) = 25

Функция evaluate\_expression(numbers, operations):

"""Вычисление значения выражения из чисел и операций"""

expression ← строка(numbers[0])

Для i от 1 до длина(numbers) - 1:

expression ← expression + операции[i-1] + строка(numbers[i])

Возвратить eval(expression)

Функция generate\_expressions(numbers):

Если длина(numbers) = 1:

Возвратить [строка(numbers[0])]

expressions ← пустой список

Для i от 1 до длина(numbers) - 1:

left\_part ← generate\_expressions(numbers[:i])

right\_part ← generate\_expressions(numbers[i:])

Для каждого left в left\_part:

Для каждого right в right\_part:

expressions.append(f"({left}+{right})")

expressions.append(f"({left}-{right})")

expressions.append(f"({left}\*{right})")

Возвратить expressions

Функция find\_target\_value(numbers, target):

operations ← ['+', '-', '\*']

all\_operations ← product(operations, repeat=длина(numbers) - 1)

Для ops в all\_operations:

expressions ← generate\_expressions(numbers)

Для каждого expr в expressions:

Попробовать:

Если eval(expr) = target:

Вывести "Найдена выражение: " + expr + " = " + target

Исключить ZeroDivisionError:

Продолжить

def evaluate\_expression(numbers, operations):

    """Вычисление значения выражения из чисел и операций"""

    expression = str(numbers[0])

    for i in range(1, len(numbers)):

        expression += operations[i-1] + str(numbers[i])

    return eval(expression)

def generate\_expressions(numbers):

    if len(numbers) == 1:

        return [str(numbers[0])]

    expressions = []

    for i in range(1, len(numbers)):

        left\_part = generate\_expressions(numbers[:i])

        right\_part = generate\_expressions(numbers[i:])

        for left in left\_part:

            for right in right\_part:

                expressions.append(f"({left}+{right})")

                expressions.append(f"({left}-{right})")

                expressions.append(f"({left}\*{right})")

    return expressions

def find\_target\_value(numbers, target):

    operations = ['+', '-', '\*']

    all\_operations = product(operations, repeat=len(numbers)-1)

    for ops in all\_operations:

        expressions = generate\_expressions(numbers)

        for expr in expressions:

            try:

                if eval(expr) == target:

                    print(f"Found expression: {expr} = {target}")

            except ZeroDivisionError:

                continue

### Четвертая задача.

### Постановка задачи

Необходимо обработать строку, содержащую перечисление любимых фильмов, и вывести на консоль следующие элементы:

• первый фильм

• последний фильм

• второй фильм

• второй фильм с конца

При этом запрещается использовать методы строк, такие как split() или find(). Нужно использовать только индексы и срезы для извлечения нужных данных.

### Решение

Для решения этой задачи можно воспользоваться следующими шагами:

1. Определение границ фильмов: Предположим, что фильмы разделены запятыми. Мы можем использовать индексы, чтобы находить начало и конец каждого фильма.

2. Извлечение фильмов: Используя срезы, мы можем получить нужные фильмы по их индексам.

3. Вывод результатов: После извлечения фильмов, мы выведем их на консоль.

ункция getInputMovie()

Ввод movies как строку от пользователя

Если длина movies больше 2

Возврат movies

Иначе

Возврат 'Терминатор, Пятый элемент, Аватар, Чужие, Назад в будущее'

Функция extractMoviesPosition(moviesString)

Создать пустой список delimeterPosition

Для каждого индекса i от 0 до длины moviesString - 1

Если символ в moviesString на позиции i равен ','

Добавить i в delimeterPosition

Возврат delimeterPosition

Функция cutString(moviesString, delimeterPosition)

Установить iter равным 0

Создать пустой список movies

Для каждого i в delimeterPosition

Добавить подстроку moviesString от iter до i в movies

Установить iter равным i + 2

Добавить подстроку moviesString от iter до конца moviesString в movies

Возврат movies

Функция showMovies(movies)

Вывести первый фильм (movies[0])

Вывести последний фильм (movies[-1])

Вывести второй фильм (movies[1])

Вывести второй фильм с конца (movies[-2])

Главная программа

Установить moviesString равным результату getInputMovie()

Вызвать showMovies(cutString(moviesString, extractMoviesPosition(moviesString)))

def getInputMovie():

    movies = input("Ввод названий фильмов через запятую: ")

    if len(movies) > 2:

        return movies

    else:

        return 'Терминатор, Пятый элемент, Аватар, Чужие, Назад в будущее'

def extractMoviesPosition(moviesString):

    delimeterPosition = []

    for i in range(len(moviesString)):

        if moviesString[i] == ',':

            delimeterPosition.append(i)

    return delimeterPosition

def cutString(moviesString, delimeterPosition):

    iter = 0

    movies = []

    for i in delimeterPosition:

        movies.append(moviesString[iter:i])

        iter = i + 2

    movies.append(moviesString[iter: len(moviesString)])

    return movies

def showMovies(movies):

    print(movies[0] + '\n',

          movies[-1] + '\n',

          movies[1] + '\n',

          movies[-2]

          )

if \_\_name\_\_ == "\_\_main\_\_":

    moviesString = getInputMovie()

    showMovies(cutString(moviesString, extractMoviesPosition(moviesString)))

## Пятая задача.

### Постановка задачи.

Создать список с членами семьи и список с их ростом. Необходимо вывести на консоль рост отца и общий рост всей семьи.

### Описание проделанной работы

**Сбор данных**: Пользователь вводит количество членов семьи и информацию о каждом: имя и рост.

**Вывод данных**: Программа показывает рост отца (если введено достаточно данных) и считает общий рост всех членов семьи

Функция inputData()

Создать пустой список my\_family

Создать пустой список my\_family\_height

Ввести количество человек в семье и сохранить в переменной num\_members

Для i от 1 до num\_members

Ввести данные в формате "NAME:HEIGHT" и сохранить в переменной data

Разделить data по символу ':' на имя и рост

Добавить имя в my\_family

Добавить (имя, рост) в my\_family\_height

Конец цикла

Вернуть my\_family\_height

Конец Функции

Функция showInfoFamily(familyHeight)

Для каждого элемента i в familyHeight

Если i[0] в нижнем регистре равно "отец"

Вывести "Рост отца - " + i[1] + " см"

Прервать цикл

Конец Если

Конец Цикла

Если цикл завершился без нахождения отца

Вывести "Отец не найден"

Конец Если

Вычислить общий рост семьи, суммируя int(member[1]) для каждого member в familyHeight

Вывести "Общий рост семьи - " + общий\_рост + " см"

Конец Функции

Если имя программы равно "main"

Вызвать showInfoFamily(inputData())

Конец Если

def inputData():

    my\_family = []

    my\_family\_height = []

    for i in range(int(input("Введите количество человек в семье: "))):

        data = input("Введите имя и рост в формате (NAME:HEIGHT): ").split(':')

        my\_family.append(data[0])

        my\_family\_height.append([data[0], data[1]])

    return my\_family\_height

def showInfoFamily(familyHeiht):

    for i in familyHeiht:

        if i[0].lower() == "отец":

            print(f"Рост отца - {i[1]} см")

            break

    else:

        print("Отец не найден")

    print(f"Общий рост семьи - {sum(int(member[1]) for member in familyHeiht)}")

if \_\_name\_\_ == "\_\_main\_\_":

    showInfoFamily(inputData())

## Шестая задача.

### Постановка

Работать со списком животных в зоопарке: добавить медведя, добавить птиц, удалить слона и вывести информацию о расположении льва и жаворонка.

### Описание проделанной работы

Программа реализует следующие шаги:

1. Вставка элемента:

• Функция insertElementAtPosition позволяет пользователю вставить новый элемент в список животных после указанного элемента.

• Сначала программа выводит текущие элементы списка. Затем запрашивает у пользователя элемент, после которого нужно вставить новый элемент.

• Если указанный элемент не найден в списке, программа сообщает об этом и завершает выполнение функции.

• Если элемент найден, новый элемент вставляется в список, и обновленный список возвращается.

2. Объединение списков:

• Функция mergeList объединяет два списка: основной список зоопарка и список птиц. Она просто возвращает новый список, который является конкатенацией двух входных списков.

3. Удаление элемента:

• Функция removeElementFromList удаляет указанный элемент из списка животных.

• Если элемент найден, он удаляется, и программа сообщает об этом. Если элемент не найден, выводится сообщение об ошибке.

4. Получение позиции элемента:

• Функция getPositionElement возвращает индекс (позицию) указанного элемента в списке. Индекс увеличивается на 1 при выводе, чтобы сделать его более понятным для пользователя (чтобы он начинался с 1 вместо 0).

5. Основная логика программы (main):

• Программа начинается с определения начального списка животных в зоопарке и списка птиц.

• Затем она запрашивает у пользователя новый элемент для вставки и обновляет список животных.

• После этого программа объединяет список зоопарка с птицами и выводит результат.

• Далее запрашивает у пользователя элемент для удаления из списка и обновляет его.

• Наконец, программа выводит позиции льва и жаворонка в списке, делая их более понятными для пользователя.

Функция insertElementAtPosition(arrayData, element)

Для каждого data в arrayData

Вывести data

Конец Цикла

Ввести элемент, после которого нужно вставить другой элемент, и сохранить в elementAfterComeNew

Если elementAfterComeNew не найден в arrayData

Вывести "Указанный элемент не найден"

Вернуться

Конец Если

Вставить element после первого вхождения elementAfterComeNew в arrayData

Вернуть arrayData

Конец Функции

Функция mergeList(baseList, mergedList)

Вернуть baseList + mergedList

Конец Функции

Функция removeElementFromList(arrayData, element)

Если element найден в arrayData

Удалить element из arrayData

Вывести "Удалено!"

Вернуть arrayData

Конец Если

Вывести "Элемент не найден"

Конец Функции

Функция getPositionElement(arrayData, element)

Вернуть индекс element в arrayData

Конец Функции

Если имя программы равно "main"

Создать список zoo = ['lion', 'kangaroo', 'elephant', 'monkey']

Создать список birds = ['rooster', 'ostrich', 'lark']

Вызвать insertElementAtPosition(zoo, ввод\_элемента) и сохранить результат в zoo

Вывести zoo

Вызвать mergeList(zoo, birds) и сохранить результат в zoo

Вывести zoo

Вызвать removeElementFromList(zoo, ввод\_элемента) и сохранить результат в zoo

Вывести zoo

Вывести "lion находится на " + (getPositionElement(zoo, "lion") + 1) + " позиции"

Вывести "lark находится на " + (getPositionElement(zoo, "lark") + 1) + " позиции"

Конец Если

# есть список животных в зоопарке

# посадите медведя (bear) между львом и кенгуру

#  и выведите список на консоль

# TODO здесь ваш код

def insertElementAtPosition(arrayData, element):

    for data in arrayData:

        print(data)

    elementAfterComeNew = input("Введите элемент, после которого нужно вставить другой элемент: ")

    if(elementAfterComeNew not in arrayData):

        print("Указанный элемент не найден")

        return

    arrayData.insert(arrayData.index(elementAfterComeNew), element)

    return arrayData

# добавьте птиц из списка birds в последние клетки зоопарка

#  и выведите список на консоль

# TODO здесь ваш код

def mergeList(baseList, mergedList):

    return baseList + mergedList

# уберите слона

#  и выведите список на консоль

# TODO здесь ваш код

def removeElementFromList(arrayData, element):

    if element in arrayData:

        arrayData.remove(element)

        print("Удалено!")

        return arrayData

    print("Элемент не найден")

# выведите на консоль в какой клетке сидит лев (lion) и жаворонок (lark).

# Номера при выводе должны быть понятны простому человеку, не программисту.

# TODO здесь ваш код

def getPositionElement(arrayData, element):

    return arrayData.index(element)

def main():

    zoo = ['lion', 'kangaroo', 'elephant', 'monkey', ]

    birds = ['rooster', 'ostrich', 'lark', ]

    zoo = insertElementAtPosition(zoo, input("Введите вставляемый элемент: "))

    print(zoo)

    zoo = mergeList(zoo, birds)

    print(zoo)

    zoo = removeElementFromList(zoo, input("Введите убираемый элемент: "))

    print(zoo)

    print(f"lion находится на {getPositionElement(zoo, "lion") + 1} позиции")

    print(f"lark находится на {getPositionElement(zoo, "lark") + 1} позиции")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

## Седьмая задача.

### Описание программы.

1. Определение списков и словарей:

• Создается список violator\_songs\_list, который содержит названия песен группы Depeche Mode и их длительности в минутах с точностью до сотых.

• Создается словарь violator\_songs\_dict, который также содержит названия песен и их длительности, но в формате ключ-значение.

2. Функции для расчета времени звучания:

• Функция calcTotalTime(songs, songsList):

• Принимает список названий песен и список песен с их длительностью.

• Суммирует длительности песен, которые есть в заданном списке, и возвращает общее время звучания, округленное до двух знаков после запятой.

• Функция calcTotalTimeForDict(songs, songsDict)

• Принимает список названий песен и словарь с их длительностью.

• Суммирует длительности песен из словаря и возвращает общее время звучания, округленное до двух знаков после запятой.

3. Основная логика:

• В функции main() создаются два списка песен.

• Вычисляется общее время звучания для первой группы песен и выводится на экран.

• Затем вычисляется общее время звучания для второй группы песен и также выводится на экран.

Определить список violator\_songs\_list как массив песен с длительностями

Определить словарь violator\_songs\_dict как пары (название песни, длительность)

Функция calcTotalTime(список песен, массив песен):

Инициализировать переменную songsTotalTime как 0.0

Для каждой песни в массиве песен:

Если название песни есть в списке песен:

Добавить длительность песни к songsTotalTime

Вернуть округленное значение songsTotalTime до 2 знаков после запятой

Функция calcTotalTimeForDict(список песен, словарь песен):

Инициализировать переменную songsTotalTime как 0.0

Для каждой песни в списке песен:

Добавить длительность песни из словаря к songsTotalTime

Вернуть округленное значение songsTotalTime до 2 знаков после запятой

Функция main():

Определить первый список песен как ['Halo', 'Enjoy the Silence', 'Clean']

Определить второй список песен как ['Sweetest Perfection', 'Policy of Truth', 'Blue Dress']

Вычислить общее время для первого списка с помощью calcTotalTime

Вывести "Три песни звучат X минут" где X - общее время

Вычислить общее время для второго списка с помощью calcTotalTimeForDict

Вывести "А другие три песни звучат Y минут" где Y - общее время

Вызвать функцию main()

# Есть список песен группы Depeche Mode со временем звучания с точностью до долей минут

# Точность указывается в функции round(a, b)

# где a, это число которое надо округлить, а b количество знаков после запятой

# более подробно про функцию round смотрите в документации https://docs.python.org/3/search.html?q=round

violator\_songs\_list = [

    ['World in My Eyes', 4.86],

    ['Sweetest Perfection', 4.43],

    ['Personal Jesus', 4.56],

    ['Halo', 4.9],

    ['Waiting for the Night', 6.07],

    ['Enjoy the Silence', 4.20],

    ['Policy of Truth', 4.76],

    ['Blue Dress', 4.29],

    ['Clean', 5.83],

]

def calcTotalTime(songs, songsList):

    songsTotalTime = 0.0

    for song in songsList:

        if(song[0] in songs):

            songsTotalTime += song[1]

    return round(songsTotalTime, 2)

def calcTotalTimeForDict(songs, songsDict):

    songsTotalTime = 0.0

    for song in songs:

        songsTotalTime += songsDict[song]

    return round(songsTotalTime, 2)

# распечатайте общее время звучания трех песен: 'Halo', 'Enjoy the Silence' и 'Clean' в формате

#   Три песни звучат ХХХ.XX минут

# Обратите внимание, что делать много вычислений внутри print() - плохой стиль.

# Лучше заранее вычислить необходимое, а затем в print(xxx, yyy, zzz)

# TODO здесь ваш код

# Есть словарь песен группы Depeche Mode

violator\_songs\_dict = {

    'World in My Eyes': 4.76,

    'Sweetest Perfection': 4.43,

    'Personal Jesus': 4.56,

    'Halo': 4.30,

    'Waiting for the Night': 6.07,

    'Enjoy the Silence': 4.6,

    'Policy of Truth': 4.88,

    'Blue Dress': 4.18,

    'Clean': 5.68,

}

# распечатайте общее время звучания трех песен: 'Sweetest Perfection', 'Policy of Truth' и 'Blue Dress'

#   А другие три песни звучат ХХХ минут

# TODO здесь ваш код

def main():

    firstListSongs = ['Halo', 'Enjoy the Silence', 'Clean']

    secondListSongs = ['Sweetest Perfection', 'Policy of Truth', 'Blue Dress']

    print(f"Три песни звучат {calcTotalTime(firstListSongs, violator\_songs\_list)} минут")

    print(f'А другие три песни звучат {calcTotalTimeForDict(secondListSongs, violator\_songs\_dict)} минут')

## Восьмая задача.

### Описание программы.

Выполняет декодирование зашифрованного сообщения, которое хранится в списке secret\_message. Каждая строка в этом списке содержит набор символов, из которых извлекаются определенные части для формирования декодированного сообщения.

Функция decrypt() выполняет следующие действия:

1. Извлекает символ из первой строки (индекс 3) и сохраняет его в переменную first\_word.

2. Извлекает подстроку из второй строки (индексы 9-12) и сохраняет её в переменную second\_word.

3. Извлекает символы из третьей строки, начиная с индекса 5 и до индекса 15, с шагом 2, и сохраняет их в переменную third\_word.

4. Извлекает подстроку из четвертой строки (индексы 7-12), разворачивает её задом наперед и сохраняет в переменную fourth\_word.

5. Извлекает подстроку из пятой строки (индексы 16-21), разворачивает её задом наперед и сохраняет в переменную fifth\_word.

6. Формирует и возвращает строку, состоящую из всех пяти переменных, разделенных пробелами.

Определить список secret\_message как массив строк

Функция decrypt():

Определить first\_word как символ из secret\_message[0] с индексом 3

Определить second\_word как подстроку из secret\_message[1] с индексами 9 до 12

Определить third\_word как подстроку из secret\_message[2] с индексами 5 до 15 с шагом 2

Определить fourth\_word как подстроку из secret\_message[3] с индексами 7 до 12, перевернутую задом наперед

Определить fifth\_word как подстроку из secret\_message[4] с индексами 16 до 21, перевернутую задом наперед

Вернуть строку, состоящую из first\_word, second\_word, third\_word, fourth\_word и fifth\_word, разделенных пробелами

secret\_message = [

    'квевтфпп6щ3стмзалтнмаршгб5длгуча',

    'дьсеы6лц2бане4т64ь4б3ущея6втщл6б',

    'т3пплвце1н3и2кд4лы12чф1ап3бкычаь',

    'ьд5фму3ежородт9г686буиимыкучшсал',

    'бсц59мегщ2лятьаьгенедыв9фк9ехб1а',

]

def decrypt():

    first\_word = secret\_message[0][3]

    second\_word = secret\_message[1][9:13]

    third\_word = secret\_message[2][5:15:2]

    fourth\_word = secret\_message[3][7:13][::-1]

    fifth\_word = secret\_message[4][16:21][::-1]

    return f"{first\_word} {second\_word} {third\_word} {fourth\_word} {fifth\_word}"

## Девятая задача.

### Постановка

В саду и на лугу сорвали цветы. Нужно создать множества цветов, произрастающих в этих двух местах, и вывести:

* Все виды цветов.
* Цветы, которые растут и там, и там.
* Цветы, которые растут в саду, но не растут на лугу.
* Цветы, которые растут на лугу, но не растут в саду.

### Описание проделанной работы

Объявить кортеж garden как ('ромашка', 'роза', 'одуванчик', 'ромашка', 'гладиолус', 'подсолнух', 'роза')

Объявить кортеж meadow как ('клевер', 'одуванчик', 'ромашка', 'клевер', 'мак', 'одуванчик', 'ромашка')

Функция flowersInSet(gardenData, meadowData):

Определить gardenSet как множество из gardenData

Определить meadowSet как множество из meadowData

Вернуть gardenSet, meadowSet

Функция flowersInfo(gardenData, meadowData):

Вывести "Цветы в саду: ", gardenData

Вывести "Цветы на лугу: ", meadowData

Вывести "Цветы, которые растут в саду и на лугу: ", gardenData & meadowData

Вывести "Цветы, которые растут в саду, но не растут на лугу: ", gardenData - meadowData

Вывести "Цветы, которые растут на лугу, но не растут в саду: ", meadowData - gardenData

gardenSet, meadowSet = flowersInSet(garden, meadow)

flowersInfo(gardenSet, meadowSet)

garden = ('ромашка', 'роза', 'одуванчик', 'ромашка', 'гладиолус', 'подсолнух', 'роза', )

# на лугу сорвали цветы

meadow = ('клевер', 'одуванчик', 'ромашка', 'клевер', 'мак', 'одуванчик', 'ромашка', )

# создайте множество цветов, произрастающих в саду и на лугу

# garden\_set =

# meadow\_set =

# TODO здесь ваш код

def flowersInSet(gardenData, meadowData):

    gardenSet = set(gardenData)

    meadowSet = set(meadowData)

    return gardenSet, meadowSet

def flowersInfo(gardenData, meadowData):

    print(f"Цветы на лугу: {gardenData}")

    print(f"Цветы на лугу: {meadowData}")

    print(f"Цветы, которые растут в саду и на лугу: {gardenData & meadowData}")

    print(f"Цветы, которые растут в саду, но не растут на лугу: {gardenData - meadowData}")

    print(f"Цветы, которые растут на лугу, но не растут в саду: {meadowData - gardenData}")

gardenSet, meadowSet = flowersInSet(garden, meadow)

flowersInfo(gardenSet, meadowSet)

## Десятая задача.

### Постановка

Есть словарь магазинов с распродажами. Нужно создать новый словарь, где для каждой сладости будут указаны два магазина с минимальными ценами.

shops = {

    'ашан':

        [

            {'name': 'печенье', 'price': 10.99},

            {'name': 'конфеты', 'price': 34.99},

            {'name': 'карамель', 'price': 45.99},

            {'name': 'пирожное', 'price': 67.99}

        ],

    'пятерочка':

        [

            {'name': 'печенье', 'price': 9.99},

            {'name': 'конфеты', 'price': 32.99},

            {'name': 'карамель', 'price': 46.99},

            {'name': 'пирожное', 'price': 59.99}

        ],

    'магнит':

        [

            {'name': 'печенье', 'price': 11.99},

            {'name': 'конфеты', 'price': 30.99},

            {'name': 'карамель', 'price': 41.99},

            {'name': 'пирожное', 'price': 62.99}

        ],

}

# Создайте словарь цен на продкты следующего вида (писать прямо в коде)

sweets = {

    'печенье': [

        {'shop': 'ашан', 'price': 10.99},

        # TODO тут с клавиатуры введите магазины и цены (можно копипастить ;)

        {'shop': 'пятерочка', 'price': 9.99}

    ],

    # TODO тут с клавиатуры введите другую сладость и далее словарь магазинов

    'конфеты': [

        {'shop': 'пятерочка', 'price': 32.99},

        {'shop': 'магнит', 'price': 30.99},

    ],

    'карамель': [

        {'shop': 'ашан', 'price': 45.99},

        {'shop': 'магнит', 'price': 41.99},

    ],

    'пирожное': [

        {'shop': 'пятерочка', 'price': 59.99},

        {'shop': 'магнит', 'price': 62.99},

    ]

}

## Одиннадцатая задача.

### Постановка

Есть словарь с товарами и их кодами, а также словарь с количеством и ценой каждого товара на складе. Цель — рассчитать общую стоимость каждого вида товара и вывести эту информацию.

### Описание проделанной работы

goods = {

    'Лампа': '12345',

    'Стол': '23456',

    'Диван': '34567',

    'Стул': '45678',

}

# Есть словарь списков количества товаров на складе.

store = {

    '12345': [

        {'quantity': 27, 'price': 42},

    ],

    '23456': [

        {'quantity': 22, 'price': 510},

        {'quantity': 32, 'price': 520},

    ],

    '34567': [

        {'quantity': 2, 'price': 1200},

        {'quantity': 1, 'price': 1150},

    ],

    '45678': [

        {'quantity': 50, 'price': 100},

        {'quantity': 12, 'price': 95},

        {'quantity': 43, 'price': 97},

    ],

}

# Рассчитать на какую сумму лежит каждого товара на складе

# например для ламп

lamps\_cost = store[goods['Лампа']][0]['quantity'] \* store[goods['Лампа']][0]['price']

# или проще (/сложнее ?)

lamp\_code = goods['Лампа']

lamps\_item = store[lamp\_code][0]

lamps\_quantity = lamps\_item['quantity']

lamps\_price = lamps\_item['price']

lamps\_cost = lamps\_quantity \* lamps\_price

print('Лампа -', lamps\_quantity, 'шт, стоимость', lamps\_cost, 'руб')

# Вывести стоимость каждого вида товара на складе:

# один раз распечать сколько всего столов и их общая стоимость,

# один раз распечать сколько всего стульев и их общая стоимость,

#   и т.д. на складе

# Формат строки <товар> - <кол-во> шт, стоимость <общая стоимость> руб

# WARNING для знающих циклы: БЕЗ циклов. Да, с переменными; да, неэффективно; да, копипаста.

# Это задание на ручное вычисление - что бы потом понять как работают циклы и насколько с ними проще жить.

# TODO здесь ваш код

tableCode = goods['Стол']

tablesQuantity = store[tableCode][0]['quantity'] + store[tableCode][1]['quantity']

tablesCost = (store[tableCode][0]['quantity'] \* store[tableCode][0]['price']

              + store[tableCode][0]['quantity'] + store[tableCode][1]['price'])

print('Стол -', tablesQuantity, 'шт, стоимость', tablesCost, 'руб')

sofa\_code = goods['Диван']

sofas\_quantity = store[sofa\_code][0]['quantity'] + store[sofa\_code][1]['quantity']

sofas\_cost = (store[sofa\_code][0]['quantity'] \* store[sofa\_code][0]['price']

              + store[sofa\_code][0]['quantity'] + store[sofa\_code][1]['price'])

print('Диван -', sofas\_quantity, 'шт, стоимость', sofas\_cost, 'руб')

chairСode = goods['Стул']

chairsQuantity = store[chairСode][0]['quantity'] + store[chairСode][1]['quantity'] + store[chairСode][2]['quantity']

chairsCost = (store[chairСode][0]['quantity'] \* store[chairСode][0]['price'] +

               store[chairСode][1]['quantity'] \* store[chairСode][1]['price'] +

               store[chairСode][2]['quantity'] \* store[chairСode][2]['price'])

print('Стул -', chairsQuantity, 'шт, стоимость', chairsCost, 'руб')

**Шпоргалка по гиту**

git init

Создает новый репозиторий.

git clone <url>

Копирует удаленный репозиторий по данному урлу на ваш компьютер.

git add <file>

Добавляет изменения в коммит. Если хотите добавить все файлы, просто используйте git add с точкой.

git commit -m "<сообщение>"

Создает новый коммит с вашими изменениями. Сообщение должно хоть-что значить в идеале.

git push

Отправляет ваши коммиты на удаленный репозиторий.

git pull

Получает изменения из удаленного репозитория и объединяет их с вашим локальным.

git rm <file>

Удаляет файл из рабочего каталога и индекса

git reset <commit>

Откатывает изменения к указанному коммиту. Без аргумента – удаляет последний коммит. С уточнением -–hard, полностью сбрасывает все изменения до последнего коммита.

**Мои контакты:**

[Zawiaha/TheoryAlgLabs](https://github.com/Zawiaha/TheoryAlgLabs)

**Телеграмм: @Chizarpik**

**Использованные материалы**

Документация Github: <https://docs.github.com/ru>

Документация Python: <https://docs.python.org/3/>