

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании
(КСУП)

ИЗМЕРЕНИЕ И ОЦЕНКА СВОЙСТВ СИСТЕМЫ. СВЁРТКА
ИЗМЕРЕНИЙ

Отчет по практической работе по дисциплине «Теория систем и
системный анализ»

Вариант 2

Выполнил

Студент гр. 513-2:

Заревич М.А.

Проверил

Ассистент каф. КСУП:

Гембух Л.А.

Томск 2025

Оглавление

ЗАДАНИЕ 1 ОБОБЩИТЬ МНЕНИЯ ЭКСПЕРТОВ: А) РАНЖИРОВАНИЕ, Б) ПАРНОЕ СРАВНЕНИЕ.....	3
ЗАДАНИЕ 2. ОБОБЩИТЬ МНЕНИЯ ЭКСПЕРТОВ, ПОЛУЧЕННЫЕ НЕПОСРЕДСТВЕННОЙ ОЦЕНКОЙ ПО БАЛЛЬНОЙ ШКАЛЕ: А) БЕЗ УЧЁТА КОМПЕТЕНТНОСТИ ЭКСПЕРТОВ, Б) С УЧЁТОМ.....	14
ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ.....	17

ЗАДАНИЕ 1 ОБОБЩИТЬ МНЕНИЯ ЭКСПЕРТОВ: А) РАНЖИРОВАНИЕ, Б) ПАРНОЕ СРАВНЕНИЕ

Задание было выполнено в Google Colaboratory, тут скриншоты программы. Сам файл загрузил вместе с отчётом.

▼ Заревич Михаил 513-2

▼ 1. Обобщить мнения экспертов:

▼ а) ранжирование

```
[325] import openpyxl
      # https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-dannymi-excel/
      # https://openpyxl.readthedocs.io/en/stable/
      # https://www.datacamp.com/community/tutorials/python-excel-tutorial
```

```
[311] workbook = openpyxl.load_workbook("/content/TS_variants_Lab_3_1-3.xlsx")
      sheet = workbook["TS_variants_Lab_3_1-3"]
```

```

▶ # список для хранения мнений экспертов
a = []

# Получаю данные из моего варианта
for row in sheet.iter_rows(min_row=11, max_row=15, values_only=True):
    a.append(row[1])
    print(row[1])

# Мой вариант row in sheet.iter_rows(min_row=11, max_row=15, values_only=True)
# Вариант Сергея for row in sheet.iter_rows(min_row=59, max_row=63, values_only=True):

```

```

⇒ A6>A1>A7>A4>A9>A5>A8>A3>A10>A2
A2>A10>A4>A5>A3>A8>A7>A1>A6>A9
A5>A4>A10>A1>A8>A6>A9>A3>A2>A7
A7>A4>A8>A2>A6>A10>A9>A3>A1>A5
A9>A2>A1>A6>A10>A8>A3>A7>A4>A5

```

```

[313] # преобразование строк в списки
for i in range(len(a)):
    a[i] = a[i].split(">")
    print(a[i])

```

```

⇒ ['A6', 'A1', 'A7', 'A4', 'A9', 'A5', 'A8', 'A3', 'A10', 'A2']
['A2', 'A10', 'A4', 'A5', 'A3', 'A8', 'A7', 'A1', 'A6', 'A9']
['A5', 'A4', 'A10', 'A1', 'A8', 'A6', 'A9', 'A3', 'A2', 'A7']
['A7', 'A4', 'A8', 'A2', 'A6', 'A10', 'A9', 'A3', 'A1', 'A5']
['A9', 'A2', 'A1', 'A6', 'A10', 'A8', 'A3', 'A7', 'A4', 'A5']

```

```

✓ [314] # Список с переменными.
0 ек. variables = ["A"+str(i+1) for i in range(len(a[0]))]
print(variables)

```

```

⇒ ['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10']

```

```

[315] # список словарей, созданный с помощью словарного и списочного выражения
# Проходится по каждому подписку, элементы каждого подписка заносит в словарь
# в котором ключи - переменные, а значение - место, выставленное экспертом.
f = [{str(a[it][i]): i+1 for i in range(len(a[it]))} for it in range(len(a))]

for i in f:
    print(i)

```

```

⇒ {'A6': 1, 'A1': 2, 'A7': 3, 'A4': 4, 'A9': 5, 'A5': 6, 'A8': 7, 'A3': 8, 'A10': 9, 'A2': 10}
{'A2': 1, 'A10': 2, 'A4': 3, 'A5': 4, 'A3': 5, 'A8': 6, 'A7': 7, 'A1': 8, 'A6': 9, 'A9': 10}
{'A5': 1, 'A4': 2, 'A10': 3, 'A1': 4, 'A8': 5, 'A6': 6, 'A9': 7, 'A3': 8, 'A2': 9, 'A7': 10}
{'A7': 1, 'A4': 2, 'A8': 3, 'A2': 4, 'A6': 5, 'A10': 6, 'A9': 7, 'A3': 8, 'A1': 9, 'A5': 10}
{'A9': 1, 'A2': 2, 'A1': 3, 'A6': 4, 'A10': 5, 'A8': 6, 'A3': 7, 'A7': 8, 'A4': 9, 'A5': 10}

```

Код для нормальной сортировки строк. Без неё он будет сортировать как A1 A10 A2

http://nedbatchelder.com/blog/200712/human_sorting.html

<https://stackoverflow.com/questions/5967500/how-to-correctly-sort-a-string-with-a-number-inside>

```
[316] # Библиотека для использования регулярных выражений
# здесь она нам нужна для поиска чисел в строке
import re
# https://skillbox.ru/media/code/regulyarnye-vyrazheniya-v-python-sintaksis-poleznye-funktsii
# https://docs.python.org/3/library/re.html

def convert(text):
    """
    Преобразует строку в число, если возможно. иначе возвращает неизменённую строку
    """
    return int(text) if text.isdigit() else text

def natural_keys(string):
    """
    Преобразует строку в список из строк и чисел в ней.

    >>> natural_keys("A10")
    ['A', 10, '']
    Получает на вход кортеж со ключом и значением

    """

    # достаёт ключ из кортежа
    text = string[0]

    # re.split(pattern, string, maxsplit=0, flags=0)
    # Split the source string by the occurrences of the pattern,
    # returning a list containing the resulting substrings.
    return [convert(c) for c in re.split(r'(\d+)', text)]
```

Сортировка словаря по ключам Это нужно для того, чтобы потом посчитать сумму рангов

```
[317] # Natural keys возвращает список в котором первое значение - буква, второе - число в виде int
# Если я всё правильно понял, то сортировка происходит следующим образом:
# Сначала сортирует по первым элементам в natural keys, то есть по буквам
# Потом по второму - по числам. Поскольку они в виде int, а не str, как были до этого
# Поэтому они будут сортироваться нормальном порядке, т.е. 1 2 ... 10
for i in range(len(f)):
    f[i] = dict(sorted(f[i].items(), key=natural_keys))

# через лямбда функцию
#f[i]= dict(sorted(f[i].items(), key = lambda x:(natural_keys(x))))
```

```
for i in f:
    print(i)
```

```
{'A1': 2, 'A2': 10, 'A3': 8, 'A4': 4, 'A5': 6, 'A6': 1, 'A7': 3, 'A8': 7, 'A9': 5, 'A10': 9}
{'A1': 8, 'A2': 1, 'A3': 5, 'A4': 3, 'A5': 4, 'A6': 9, 'A7': 7, 'A8': 6, 'A9': 10, 'A10': 2}
{'A1': 4, 'A2': 9, 'A3': 8, 'A4': 2, 'A5': 1, 'A6': 6, 'A7': 10, 'A8': 5, 'A9': 7, 'A10': 3}
{'A1': 9, 'A2': 4, 'A3': 8, 'A4': 2, 'A5': 10, 'A6': 5, 'A7': 1, 'A8': 3, 'A9': 7, 'A10': 6}
{'A1': 3, 'A2': 2, 'A3': 7, 'A4': 9, 'A5': 10, 'A6': 4, 'A7': 8, 'A8': 6, 'A9': 1, 'A10': 5}
```

Поиск суммы рангов

```
[318] # список для хранения значений
# 10 значений для всех 10 переменных
temp = [0 for i in range(len(a[0]))]

count = 0

for i in f:
    count = 0

    # ранг каждой переменной заносит на своё место в список temp
    for value in i.values():
        temp[count] += value
        count += 1
```

```
SummaRangov = dict(zip(variables, temp))
print(SummaRangov)

# сортировка словаря по возрастанию значений
SummaRangov = dict(sorted(SummaRangov.items(), key = lambda x:(x[1])))
print(SummaRangov)

{'A1': 26, 'A2': 26, 'A3': 36, 'A4': 20, 'A5': 31, 'A6': 25, 'A7': 29, 'A8': 27, 'A9': 30, 'A10': 25}
{'A4': 20, 'A6': 25, 'A10': 25, 'A1': 26, 'A2': 26, 'A8': 27, 'A7': 29, 'A9': 30, 'A5': 31, 'A3': 36}
```

Поиск дубликатов, т.е. связанных рангов

```
[319] from collections import defaultdict

# словарь с дубликатами
Duplicates = defaultdict(list)

for key,value in SummaRangov.items():
    Duplicates[value].append(key)

Duplicates = {key:value for key,value in Duplicates.items() if len(value)>1}

# ключи - повторяющиеся значения в словаре SummaRangov. значения - ключи, с этими значениями
print(Duplicates)
```

```
{25: ['A6', 'A10'], 26: ['A1', 'A2']}
```

Присваивание рангов

```
[322] # Списки с ключами и значениями
keys = list(SummaRangov.keys())
values = list(SummaRangov.values())

count = 1
i = 0

# цикл по элементам списка с ключами
while i < len(keys):

    # Если его значения нет как ключа в списке дубликатов, присваивает этому ключу его ранг
    if values[i] not in Duplicates:
        SummaRangov[keys[i]] = count
        count+=1
        i+=1
        continue

    # Если значение есть как ключ в списке дубликатов
    if values[i] in Duplicates:

        # количество ключей с одинаковыми значениями - длина списка
        length = len(Duplicates[values[i]])

        # каждому ключу, у которых дублируется данное значение, меняет значение на среднее значение присваивает
        for j in Duplicates[values[i]]:
            SummaRangov[j] = (2*count+length-1)/length

        # каждому ключу, у которых дублируется данное значение, меняет значение на среднее значение присваивает
        for j in Duplicates[values[i]]:
            SummaRangov[j] = (2*count+length-1)/length

        # удаляет просмотренные ключи из словаря
        del Duplicates[values[i]]

        # увеличивает i и count на количество дубликатов, чтобы пропустить их и не заносить им место
        i+=length
        count += length
```

Данные были получены по ранговой шкале обратного порядка. То есть, элемент A4 - самый худший, а A4 - самый лучший

```
[323] print("Обобщённый ранг: ")
print(SummaRangov)
```

Обобщённый ранг:
{ 'A4': 1, 'A6': 2, 'A10': 3, 'A1': 4, 'A2': 5, 'A8': 6, 'A7': 7, 'A9': 8, 'A5': 9, 'A3': 10 }

✓ б) парное сравнение

✓ Матрицы парных сравнений

```
# Список из 5 матриц парных сравнений, по 1 матрице для каждого эксперта
# у каждой матрицы размер 10 на 10.
ListOfMatrixes = [[0 for k in range(len(a[0])) for j in range(len(a[0])) for i in range(len(f))]
for i in ListOfMatrixes:
    for j in i:
        print(j)
    print()
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```



```

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

```

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

```

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

```

[327] for matrix in range(len(ListOfMatrixes)):

    # ранги текущего эксперта(первый эксперт - первая матрица, второй-вторая и т.д.)
    CurrentDict = f[matrix]

    for row in range(len(ListOfMatrixes[matrix])):

        #
        Keys = list(CurrentDict.keys())
        Values = list(CurrentDict.values())

        # текущие ключ и значение. соответствуют номеру столбца
        currentkey = int(Keys[row][1:])
        currentvalue = Values[row]

        for column in range(len(ListOfMatrixes[matrix][row])):
            # если значение элемента матрицы больше значения текущего ключа, заносит 1. иначе 0
            if Values[column]>=currentvalue:
                ListOfMatrixes[matrix][row][column] = 1
            else:
                ListOfMatrixes[matrix][row][column] = 0

```

```

✓ [328] for i in f:
0      print(i)
сек.

```

```

{'A1': 2, 'A2': 10, 'A3': 8, 'A4': 4, 'A5': 6, 'A6': 1, 'A7': 3, 'A8': 7, 'A9': 5, 'A10': 9}
{'A1': 8, 'A2': 1, 'A3': 5, 'A4': 3, 'A5': 4, 'A6': 9, 'A7': 7, 'A8': 6, 'A9': 10, 'A10': 2}
{'A1': 4, 'A2': 9, 'A3': 8, 'A4': 2, 'A5': 1, 'A6': 6, 'A7': 10, 'A8': 5, 'A9': 7, 'A10': 3}
{'A1': 9, 'A2': 4, 'A3': 8, 'A4': 2, 'A5': 10, 'A6': 5, 'A7': 1, 'A8': 3, 'A9': 7, 'A10': 6}
{'A1': 3, 'A2': 2, 'A3': 7, 'A4': 9, 'A5': 10, 'A6': 4, 'A7': 8, 'A8': 6, 'A9': 1, 'A10': 5}

```

0 BK.

↑ ↓ ↗ ↶ ↷ ⚙

▶

```
for i in ListOfMatrixes:
    for j in i:
        print(j)
    print()
```

⇌

[1, 1, 1, 1, 1, 0, 1, 1, 1, 1]

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 1, 1, 0, 0, 0, 0, 0, 0, 1]

[0, 1, 1, 1, 1, 0, 0, 1, 1, 1]

[0, 1, 1, 0, 1, 0, 0, 1, 0, 1]

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

[0, 1, 1, 1, 1, 0, 1, 1, 1, 1]

[0, 1, 1, 0, 0, 0, 0, 1, 0, 1]

[0, 1, 1, 0, 1, 0, 0, 1, 1, 1]

[0, 1, 0, 0, 0, 0, 0, 0, 0, 1]

[1, 0, 0, 0, 0, 1, 0, 0, 1, 0]

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

[1, 0, 1, 0, 0, 1, 1, 1, 1, 0]

[1, 0, 1, 1, 1, 1, 1, 1, 1, 0]

[1, 0, 1, 0, 1, 1, 1, 1, 1, 0]

[0, 0, 0, 0, 0, 1, 0, 0, 1, 0]

[1, 0, 0, 0, 0, 1, 1, 0, 1, 0]

[1, 0, 0, 0, 0, 1, 1, 1, 1, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

[1, 0, 1, 1, 1, 1, 1, 1, 1, 1]

[1, 1, 1, 0, 0, 1, 1, 1, 1, 0]

[0, 1, 0, 0, 0, 0, 1, 0, 0, 0]

[0, 1, 1, 0, 0, 0, 1, 0, 0, 0]

[1, 1, 1, 1, 0, 1, 1, 1, 1, 1]

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

[0, 1, 1, 0, 0, 1, 1, 0, 1, 0]

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

[0, 1, 1, 0, 0, 1, 1, 1, 1, 0]

→

```
[1, 1, 1, 0, 0, 1, 1, 1, 1, 1]
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 1, 1, 0, 0, 1, 1]
[1, 0, 1, 0, 1, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 0, 0, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 0, 1, 1, 1]
[1, 0, 1, 0, 1, 0, 0, 0, 1, 0]
[1, 0, 1, 0, 1, 0, 0, 0, 1, 1]

[1, 0, 1, 1, 1, 1, 1, 1, 0, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 0, 1]
[0, 0, 1, 1, 1, 0, 1, 0, 0, 0]
[0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 0, 1]
[0, 0, 0, 1, 1, 0, 1, 0, 0, 0]
[0, 0, 1, 1, 1, 0, 1, 1, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[0, 0, 1, 1, 1, 0, 1, 1, 0, 1]
```

✓ Обобщённая матрица

```
[330] # Обобщённая матрица
A = [[0 for j in range(len(f[0]))] for i in range(len(f[0]))]
for i in A:
    print(i)
```

```

⇌ [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

```

[331] # заполнение обобщённой матрицы

# range 1 т.к. по всем матрицам нужно проходиться не в этом цикле
for matrix in range(1):

    for row in range(len(ListOfMatrixes[matrix])):
        for column in range(len(ListOfMatrixes[matrix][row])):

            # счётчики нулей и единиц
            onecount = 0
            zerocount = 0

            # ищет нули и единицы в элементе [row][column] остальных матриц
            for k in range(len(ListOfMatrixes)):
                if ListOfMatrixes[k][row][column] == 1:
                    onecount += 1
                else:
                    zerocount += 1

```

```

    if onecount - zerocount >=0:
        A[row][column] = 1
    else:
        A[row][column] = 0

```

Вывод обобщённой матрицы и рангов, полученных с помощью неё

```

[201] for i in A:
      print(i)

      print()
      ranks = [0 for i in range(len(A))]
      for i in range(len(A)):
          for j in range(len(A[i])):
              ranks[i] += A[i][j]
      print(ranks)

```

```

⇒ [1, 0, 1, 0, 1, 1, 1, 1, 1, 0]
   [1, 1, 1, 0, 1, 1, 1, 0, 0, 1]
   [0, 0, 1, 0, 0, 0, 1, 0, 0, 0]
   [1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
   [0, 0, 1, 0, 1, 0, 0, 1, 0, 0]
   [0, 0, 1, 0, 1, 1, 1, 0, 1, 1]
   [0, 0, 0, 1, 1, 0, 1, 0, 1, 0]
   [0, 1, 1, 0, 0, 1, 1, 1, 1, 0]
   [0, 1, 1, 0, 1, 0, 0, 0, 1, 0]
   [1, 0, 1, 0, 1, 0, 1, 1, 1, 1]

   [7, 7, 2, 9, 3, 6, 4, 6, 4, 7]

```

ЗАДАНИЕ 2. ОБОБЩИТЬ МНЕНИЯ ЭКСПЕРТОВ, ПОЛУЧЕННЫЕ
НЕПОСРЕДСТВЕННОЙ ОЦЕНКОЙ ПО БАЛЛЬНОЙ ШКАЛЕ: А) БЕЗ
УЧЁТА КОМПЕТЕНТНОСТИ ЭКСПЕРТОВ, Б) С УЧЁТОМ.

Задание было выполнено в Google Colaboratory, тут скриншоты программы.
Сам файл загрузил вместе с отчётом.

✓ 2. Обобщить мнения экспертов, полученные
непосредственной оценкой по балльной шкале:

✓ а) без учёта компетентности экспертов,

```
✓ [265] # мнения экспертов
0      ranks = []
сек.    # компетентность экспертов
        coefficients = []

        # Получаю данные из моего варианта
        for row in sheet.iter_rows(min_row=11, max_row=15, values_only=True):
            ranks.append(list(row[7:17]))
            coefficients.append(float(row[18]))
        for i in ranks:
            print(i)
        print()
        print(coefficients)
```

```
⇒ [7, 1, 9, 10, 7, 8, 8, 4, 7, 2]
   [3, 7, 7, 2, 2, 5, 10, 4, 6, 3]
   [4, 9, 6, 6, 10, 3, 8, 8, 4, 6]
   [5, 1, 3, 10, 2, 9, 6, 10, 1, 5]
   [9, 7, 4, 6, 5, 1, 3, 2, 2, 3]

   [0.128642, 0.12964, 0.15756, 0.034818, 0.54934]
```

```
[ ] # транспонирует матрицу.
# не понимаю, как это работает.
for i in zip(*ranks):
    print(i)
```

```
⇒ (7, 3, 4, 5, 9)
(1, 7, 9, 1, 7)
(9, 7, 6, 3, 4)
(10, 2, 6, 10, 6)
(7, 2, 10, 2, 5)
(8, 5, 3, 9, 1)
(8, 10, 8, 6, 3)
(4, 4, 8, 10, 2)
(7, 6, 4, 1, 2)
(2, 3, 6, 5, 3)
```

```
[276] FinalRanks= []
```

```
for i in zip(*ranks):
    count = 0
    for j in i:
        count+=j

    FinalRanks.append(count/5)
```

```
✓ [334] FinalRanksWoCoef = dict(zip(variables, FinalRanks))
0
:ЕК. for i in FinalRanksWoCoef.items():
    print(i)
```

```
⇒ ('A1', 7.037804)
('A2', 6.33436)
('A3', 5.312432)
('A4', 6.135280000000001)
('A5', 5.55171)
('A6', 3.012718)
('A7', 5.442944000000001)
('A8', 3.740468)
('A9', 3.4420720000000005)
('A10', 3.4136740000000003)
```

✓ б) с учётом компетентности

```
✓ [306] FinalRanks = []
0
:ЕК. for i in zip(*ranks):
    count = 0
    for j in range(len(i)):
        count+=i[j]*coefficients[j]

    FinalRanks.append(count)
```

```
✓ [333] FinalRanksWCoef = dict(zip(variables, FinalRanks))  
J  
HK.     for i in FinalRanksWCoef.items():  
         print(i)
```

```
⇒ ('A1', 7.037804)  
   ('A2', 6.33436)  
   ('A3', 5.312432)  
   ('A4', 6.1352800000000001)  
   ('A5', 5.55171)  
   ('A6', 3.012718)  
   ('A7', 5.4429440000000001)  
   ('A8', 3.740468)  
   ('A9', 3.4420720000000005)  
   ('A10', 3.4136740000000003)
```


ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем состоит метод ранжирования? Как обрабатываются результаты группового ранжирования?

Метод ранжирования предполагает упорядочивание объектов экспертом от наиболее предпочтительного к наименее предпочтительному. Обработка результатов группового ранжирования происходит следующим образом: для каждого объекта суммируются ранги, присвоенные ему каждым экспертом. Объекты сортируются по возрастанию суммы рангов. Объектам присваиваются обобщённые ранги в соответствии с их позицией в полученном порядке.

2. Опишите метод парных сравнений, а также процедуру построения обобщенной матрицы парных сравнений.

Метод парных сравнений заключается в сравнении всех возможных пар объектов и определении предпочтения для каждой пары. Создаётся матрица, где строки и столбцы соответствуют объектам. Для каждой пары объектов (i , j) подсчитывается количество экспертов, которые предпочли объект i объекту j . Если объект i предпочтительнее объекта j более чем половиной экспертов, то в ячейку (i , j) матрицы записывается 1. В противном случае записывается 0.

3. В чем состоит метод непосредственной оценки? Как обрабатываются данные групповой экспертизы?

Метод непосредственной оценки заключается в подсчёте оценок экспертов по одному объекту и поиска среднего арифметического по сумме оценок от экспертов. Обработка данных без учёта компетентности экспертов: все оценки по объекту суммируются и делятся на количество экспертов. С учётом компетентности экспертов: учитывается компетентность экспертов. Каждому эксперту назначается вес, и оценка каждого эксперта умножается на его вес перед суммированием.

4. В чем заключается метод последовательного сравнения (Черчмена – Акоффа)?

Данный метод сочетает в себе ранжирование и непосредственную оценку.

Осуществляется ранжирование объектов. После этого проводят непосредственную оценку объектов. Эксперт решает, будет ли первый объект

по важности превосходить все предыдущие. Если да, к нему добавляется значение, чтобы его ранг был выше суммы всех остальных. Если нет, ранг меняется так, чтобы он был меньше суммы остальных. Затем анализируется следующий объект.