5. Золотоносный участок. План золотоносного участка, имеющего прямоугольную форму размерностью ***n*m (n, m ≤ 50)***, разделен на квадратные зоны с длиной стороны 1 (см. рис. 1). Каждая зона содержит некоторое количество золота определенного качества.

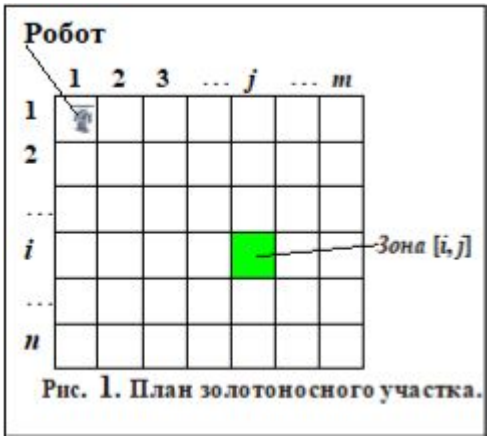


Рис. 1. План золотоносного участка.

Необходимая информация о зонах записана в следующие два файла:

. текстовый файл Aur.in содержит в первой строке натуральные числа n и m, отделенные пробелом. Каждая из следующих n строк данного файла содержит m вещественных чисел, разделенных пробелом – элементы матрицы A; каждый из которых представляет собой количество ai,j грамм золота которое можно добыть в зоне с координатами [i, j];

. текстовый файл Calitate.in, каждая из n строк которого содержит m цифр из множества {1,2,3,4}, отделенных одним пробелом, каждая из которых представляет собой качество золота в зоне [i, j], $1 \le i \le n$, $1 \le j \le m$.

Разработать приложение, которое с помощью меню и подпрограмм выполняет по выбору пользователя следующие действия:

1) Дописывает к плану участка новую строку/новый столбец; строка или столбец и их позиция должны вводится с клавиатуры;

2) Удаляет из плана участка граничную строку/столбец; выбор строки(северной/южной) или столбца (западного/восточного) должны вводится с клавиатуры;

3) Определяет координаты (номер строки и номер столбца) участков с локально минимальной высотой, т. е. координаты не граничных участков с количеством золота, строго меньшей, чем количество золота из максимум 8 соседних зон);

4) Определяет среднее количество золота с участков для каждого типа качества используемого в файле Calitate.in;

5) Выводит на экран список кодов качества золота в убывающем порядке по количеству зон соответствующего качества; сортировку выполнить методом быстрой сортировки;

6) Создаёт текстовый файл ConstCal.txt, который будет содержать строки входного файла Aur.in, которые содержат золото только первого и второго качества;

7) Находит прямоугольник максимальной площади, который содержит по углам только золото качества CodC; значение CodC (одно из чисел 1, 2, 3 или 4) вводится с клавиатуры; стороны прямоугольника совпадают со сторонами зон на плане. На экран выводится площадь S, координаты левого нижнего и правого верхнего угла найденного прямоугольника;

8) Решите задачу: В зоне, расположенной в северо-западном углу, находится робот. Из зоны с координатами [i, j], $1 \le i \le n$, $1 \le j \le m$, робот может добыть не более ai,j граммов золота. – значения записаны в файл Aur.in. По технологическим соображениям, на участке существуют ограничения: на каждом шаге робот может перемещаться из текущей зоны только в одну из соседних зон восточнее или южнее.

Напишите программу, которая находит максимальное количество золота Cmn которое может добыть робот, независимо от качества, а также один из маршрутов, обеспечивающих добычу такого количества золота.


Рис.2. *Посещенные зоны*

Входные данные. Необходимая об участке информация записана в текстовом файле Aur.in, описанном ранее.

Выходные данные. На экран выводятся вещественное число Cmn и найденный путь, представленный координатами соответствующих участков.

Например, для рисунка 2, пройденный путь будет следующим:

[1, 1]–[2, 1]–[2, 2]–[3, 2]–[4, 2]–[4, 3]–[4, 4]–[5, 4]–[6, 4]–[6, 5]–[6, 6]–

[6, 7]–[6, 8] –[7, 8]

Aur.in

```
5 5
753 352 732 441 235
212 387 485 210 243
495 482 791 370 714
788 759 535 356 112
140 309 250 299 494
```

Calitate.in

```
2 2 3 1 2
4 4 2 4 4
1 1 2 1 4
2 1 2 2 3
3 1 4 3 1
```

День 1

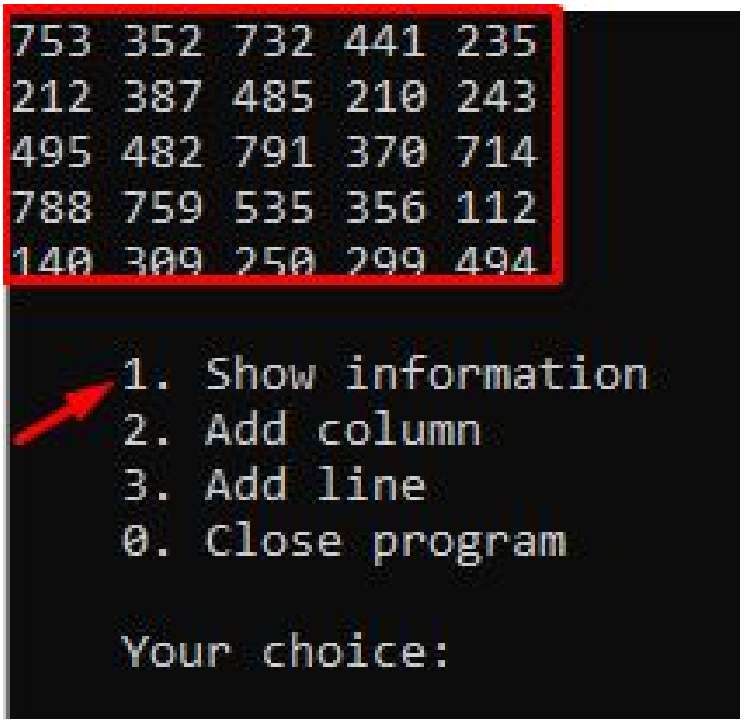
```
def show_file():
   with open('Aur.in', 'r') as file:
        lst = file.readlines()
   del lst[0]
   lst = [[int(n) for n in x.split()] for x in lst]
   print(lst)

while True:
   print('''
   1. Show information
   2. Close program

   Your choice:
   ''')
   x  = int(input())
   if x == 1: show_file()
   elif x == 1: break
   else: print('Incorrect number')
```

Библиография

- https://labs-org.ru/python-9/
- https://labs-org.ru/python-8/
- https://www.youtube.com/watch?v=W8KRzm-HUcc&t=243s

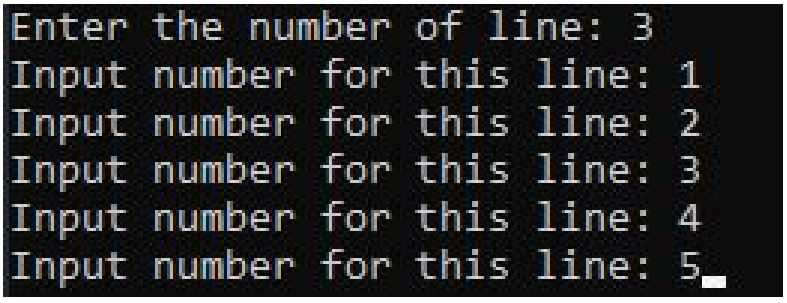
День 2

```
def add_column():
    k = int(input('Enter the number of column: '))

    for n in matrix:
        n.insert(k - 1, 0)

    dataframe = pd.DataFrame(data=matrix)
    dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

    f = open('Aur.in','r+')
    li = f.readlines() # read old content
    f.seek(0) # go back to the beginning of the file
    f.write(str(lines) + ' ')
    f.write(str(columns) + '\n')# write new content at the beginning
    for line in li: # write old content after new
        f.write(line)

def add_line():
    mat = []
    k = int(input('Enter the number of line: '))
    for i in range(int(line)):
        a = int(input('Input number for this line: '))
        mat.append(a)

    matrix.insert(k - 1, mat)

    dataframe = pd.DataFrame(data=matrix)# Reading information from user
    dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)# Writing
this information in file

    f = open('Aur.in','r+')
    lines = f.readlines() # read old content
    f.seek(0) # go back to the beginning of the file
    add_line = int(list_number[0]) + 1
    f.write(str(add_line) + ' ')
    f.write(str(column) + '\n')# write new content at the beginning
    for f_line in lines: # write old content after new
        f.write(f_line)
```




```
# Вывод матрицы из файла
```


```
# Добавляет столбец со значениями во всех строках = 0
```




```
# Выбираем положение строки и потом вводим значения для каждого элемента этой строки
```

Библиография

- https://yandex.ru/turbo?text=https%3A%2F%2Finternet-technologies.ru%2Farticles%2Fobektno-orientirovannoe-programmirovanie-v-python.html
- https://docs.scipy.org/doc/numpy/reference/maskedarray.generic.html#

День 3

```
def del_line():
    file = open('Aur.in', 'r')
    array = file.readline().split()
    list_number = [num for num in array]
    line = list_number[0]
    column = list_number[1]
    array2 = [list(map(int, row.split())) for row in file.readlines()]
    matrix = array2

    print('Current count of lines is: ' + str(line))
    k = int(input('Enter the number of line which you want to delete: '))

    matrix.remove(matrix[k - 1])

    dataframe = pd.DataFrame(data=matrix)
    dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

    f = open('Aur.in','r+')
    lines = f.readlines() # read old content
    f.seek(0) # go back to the beginning of the file
    del_line = int(line) - 1
    f.write(str(del_line) + ' ')
    f.write(str(column) + '\n')# write new content at the beginning
    for f_line in lines: # write old content after new
        f.write(f_line)
    cls()

def del_column():
    file = open('Aur.in', 'r')
    array = file.readline().split()
    list_number = [num for num in array]
    line = list_number[0]
    column = list_number[1]
    array2 = [list(map(int, row.split())) for row in file.readlines()]
    matrix = array2

    print('Current count of column is: ' + str(column))
    k = int(input('Enter the number of line which you want to delete: '))
```

```
matrix = np.delete(matrix, k - 1, axis=1)

dataframe = pd.DataFrame(data=matrix)
dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

f = open('Aur.in','r+')
lines = f.readlines() # read old content
f.seek(0) # go back to the beginning of the file
del_column = int(column) - 1

f.write(str(line) + ' ')
f.write(str(del_column) + '\n')# write new content at the beginning
for f_line in lines: # write old content after new
    f.write(f_line)
cls()
```

Библиография

- https://askdev.ru/q/kak-udalit-stolbcy-v-numpy-matrica-65296/

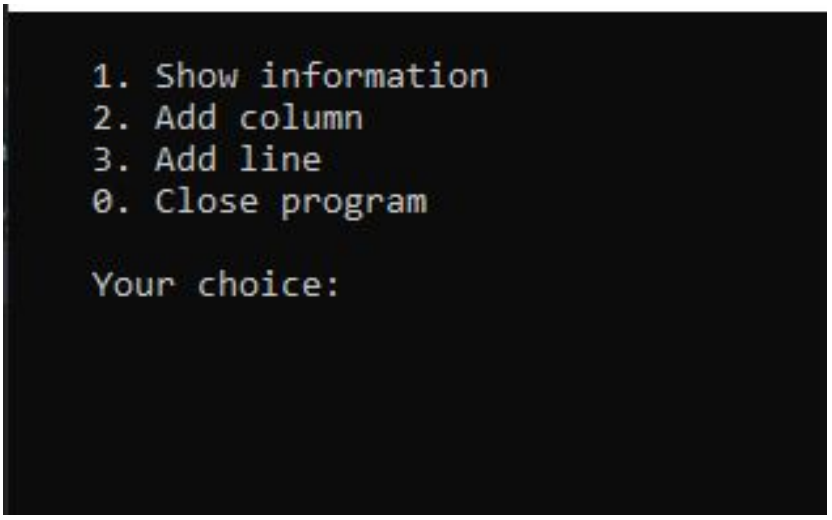
День 4

```
def del_column():
    file = open('Aur.in', 'r')
    array = file.readline().split()
    list_number = [num for num in array]
    line = list_number[0]
    column = list_number[1]
    array2 = [list(map(int, row.split())) for row in file.readlines()]
    matrix = array2

    if column != '0' and line != '0':
        print('Current count of column is: ' + str(column))
        k = int(input('Enter the number of line which you want to delete: '))

        while k < 1 or k > int(column):
            cls()
            print('This number of line doesn\'t exist: ')
            k = int(input('Enter the number of line which you want to delete:
'))

        matrix = np.delete(matrix, k - 1, axis=1)

        dataframe = pd.DataFrame(data=matrix)
        dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

        f = open('Aur.in','r+')
        lines = f.readlines() # read old content
        f.seek(0) # go back to the beginning of the file
        del_column = int(column) - 1
        if del_column == 0:
            line = 0
        f.write(str(line) + ' ')
        f.write(str(del_column) + '\n')# write new content at the beginning
        for f_line in lines: # write old content after new
            f.write(f_line)

        cls()
    else:
        print('\n    File is empty!\n')
```

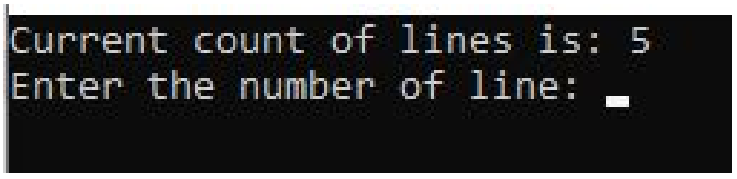
```
def del_line():
    file = open('Aur.in', 'r')
    array = file.readline().split()
    list_number = [num for num in array]
    line = list_number[0]
    column = list_number[1]
    array2 = [list(map(int, row.split())) for row in file.readlines()]
    matrix = array2

    if column != '0' and line != '0':
        print('Current count of lines is: ' + str(line))
        k = int(input('Enter the number of line which you want to delete: '))

        while k < 1 or k > int(column):
            cls()
            print('This number of line doesn\'t exist: ')
            k = int(input('Enter the number of line which you want to delete:
'))
        matrix.remove(matrix[k - 1])

        dataframe = pd.DataFrame(data=matrix)
        dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

        f = open('Aur.in','r+')
        lines = f.readlines() # read old content
        f.seek(0) # go back to the beginning of the file
        del_line = int(line) - 1
        if del_line == 0:
            column = 0
        f.write(str(del_line) + ' ')
        f.write(str(column) + '\n')# write new content at the beginning
        for f_line in lines: # write old content after new
            f.write(f_line)
        cls()
    else:
        print('\n    File is empty!\n')
```

```
def add_line():
    file = open('Aur.in', 'r')
    array = file.readline().split()
    list_number = [num for num in array]
    line = list_number[0]
    column = list_number[1]
    array2 = [list(map(int, row.split())) for row in file.readlines()]
    matrix = array2
    mat = []
    print('Current count of lines is: ' + str(line))
    k = int(input('Enter the position of the line where it will be placed: '))
    cls()
    answer = input('Do you want to enter numbers manually?(y/n): ')
    if answer == 'n':
        cls()
        a = int(input('Enter the first border range: '))
        b = int(input('Enter the second border range: '))
        mat = [random.randint(a, b) for i in range(int(column))]
    else:
        cls()
        print('Input numbers for this line: ')
        for i in range(int(column)):
            print('[' + str(i + 1) + ']', end='')
            a = int(input('--> '))
            mat.append(a)

    matrix.insert(k - 1, mat)
    dataframe = pd.DataFrame(data=matrix)
    dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)
    f = open('Aur.in','r+')
    lines = f.readlines() # read old content
    f.seek(0) # go back to the beginning of the file
    add_line = int(line) + 1
    f.write(str(add_line) + ' ')
    f.write(str(column) + '\n')# write new content at the beginning
    for f_line in lines: # write old content after new
        f.write(f_line)
    cls()
```

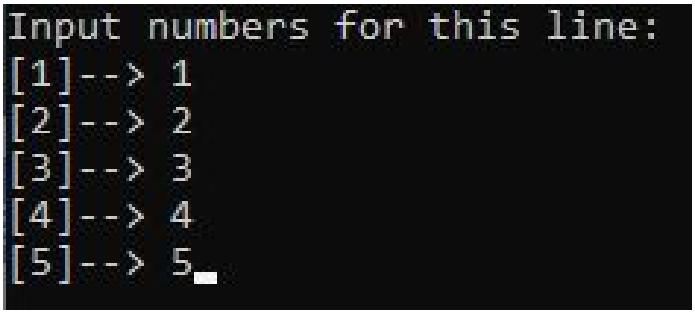
```
def add_column():
    file = open('Aur.in', 'r')
    array = file.readline().split()
    list_number = [num for num in array]
    line = list_number[0]
    column = list_number[1]
    array2 = [list(map(int, row.split())) for row in file.readlines()]
    matrix = array2

    print('Current count of columns is: ' + str(column))
    k = int(input('Enter the position of the column where it will be placed:
'))
    cls()

    answer = input('Do you want to enter numbers manually?(y/n): ')
    if answer == 'n':
        cls()
        a = int(input('Enter the first border range: '))
        b = int(input('Enter the second border range: '))
        for i in matrix:
            i.insert(k - 1, random.randint(a, b))
    else:
        cls()
        print('Input numbers for this column: ')
        for i in matrix:
            i.insert(k - 1, input('==>'))

    dataframe = pd.DataFrame(data=matrix) # Reading information
    dataframe.to_csv('Aur.in', sep=' ', header=False, index=False) # Writing
information in file

    f = open('Aur.in','r+')
    lines = f.readlines() # Read old content
    f.seek(0) # Go back to the beginning of the file
    add_column = int(column) + 1
    f.write(str(line) + ' ')
    f.write(str(add_column) + '\n') # Write new content at the beginning
    for f_line in lines: # Write old content after new
        f.write(f_line)
    column = add_column
    cls()
```



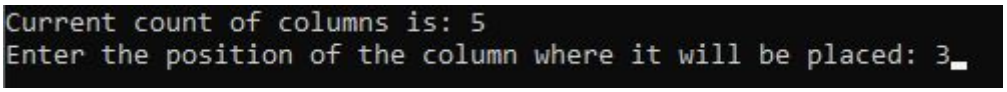
Меню программы



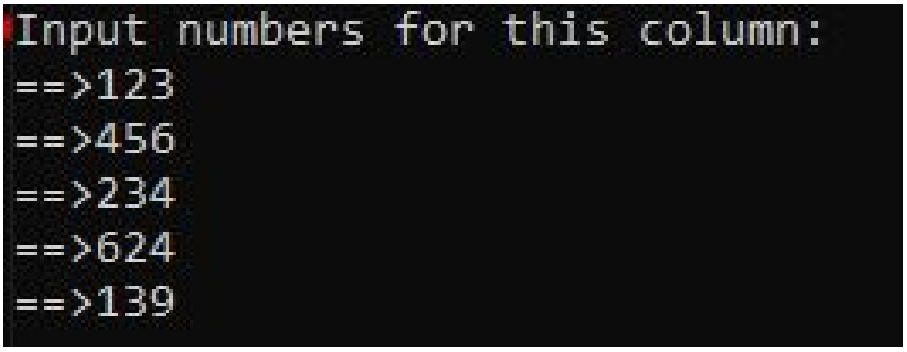
Добавление строки 1 часть(Выбор положения строки)



Добавление строки 2 часть(Ввод данных в строку)



Добавление столбца 1 часть(Выбор положения столбца)



Добавление столбца 2 часть(Ввод данных в столбец)

Библиография

- https://otus.ru/nest/post/522/

День 5

```
def add_line():
   mat = []
   m_q = []
   m = []
   l = 0
   c = 0


   matrix_q = get_calitate(mat)
   matrix_a, line, column = get_aur(mat, l, c)


   waiting = input('If you want to go to the main menu enter \'quit\'.\nIf you
want co continue enter \'go\': ')
   if waiting == 'go':
        cls()
        if line == '0' and column == '0':
             print('Current count of lines is: ' + str(line))
             new_column = int(input('Enter count of numbers for this column:
'))
             cls()


             answer = input('Do you want to enter numbers manually?(y/n): ')
             if answer == 'n':
             cls()
             a = int(input('Enter the first border range: '))
             b = int(input('Enter the second border range: '))
             m = [random.randint(a, b) for i in range(int(new_column))]
             m_q = [random.randint(1, 4) for i in range(int(new_column))]
             else:
             cls()
             print('Input numbers for this line: ')
             for i in range(int(new_column)):
                  print('[' + str(i + 1) + ']', end='')
                  a = int(input('==> '))
                  m.append(a)
             for i in range(int(new_column)):
                  print('[' + str(i + 1) + ']', end='')
                  a = int(input('==> '))
```

```
                m_q.append(a)

            matrix_a.append(m)
            matrix_q.appent(m_q)

            dataframe = pd.DataFrame(data=matrix_a)
            dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

            dataframe = pd.DataFrame(data=matrix_q) # Reading information
            dataframe.to_csv('Calitate.in', sep=' ', header=False,
index=False) # Writing information in file

            f = open('Aur.in','r+')
            lines = f.readlines() # read old content
            f.seek(0) # go back to the beginning of the file
            column = new_column
            add_line = int(line) + 1
            f.write(str(add_line) + ' ')
            f.write(str(column) + '\n')# write new content at the beginning
            for f_line in lines: # write old content after new
            f.write(f_line)

        else:
            print('Current count of lines is: ' + str(line))
            k = int(input('Enter the position of the line where it will be
placed: '))
            cls()

            answer = input('Do you want to enter numbers manually?(y/n): ')
            if answer == 'n':
            cls()
            a = int(input('Enter the first border range: '))
            b = int(input('Enter the second border range: '))
            m = [random.randint(a, b) for i in range(int(column))]
            m_q = [random.randint(1, 4) for i in range(int(column))]
            else:
            cls()
            print('Input numbers for this line: ')
            for i in range(int(column)):
                print('[' + str(i + 1) + ']', end='')
                a = int(input('==> '))
```

```
                m.append(a)
            print('\nEnter gold quality(1 - 4): ')
            for i in range(int(column)):
                print('[' + str(i + 1) + ']', end='')
                a = int(input('==> '))
                m_q.append(a)

            matrix_q.insert(k - 1, m_q)
            matrix_a.insert(k - 1, m)

            dataframe = pd.DataFrame(data=matrix_a)
            dataframe.to_csv('Aur.in', sep=' ', header=False, index=False)

            dataframe = pd.DataFrame(data=matrix_q) # Reading information
            dataframe.to_csv('Calitate.in', sep=' ', header=False,
index=False) # Writing information in file

            f = open('Aur.in','r+')
            lines = f.readlines() # read old content
            f.seek(0) # go back to the beginning of the file
            add_line = int(line) + 1
            f.write(str(add_line) + ' ')
            f.write(str(column) + '\n')# write new content at the beginning
            for f_line in lines: # write old content after new
            f.write(f_line)
        cls()
        print(Color('{green}Information successfully written!{/green}\n'))
    else:
        pass
```

Библиография

–

День 6

```
def average():
   mat = []
   mat_1 = []
   mat_2 = []
   mat_3 = []
   mat_4 = []
   l = 0
   c = 0
   matrix_a, line, column = get_aur(mat, l, c)
   matrix_q = get_calitate(mat)
   for i in range(int(line)):
        for j in range(int(column)):
             if matrix_q[i][j] == 1:
             mat_1.append(matrix_a[i][j])
             if matrix_q[i][j] == 2:
             mat_2.append(matrix_a[i][j])
             if matrix_q[i][j] == 3:
             mat_3.append(matrix_a[i][j])
             if matrix_q[i][j] == 4:
             mat_4.append(matrix_a[i][j])

   average_1 = sum(mat_1) / len(mat_1)
   print('Average amount of gold with quality 1:',  round(average_1, 2))

   average_2 = sum(mat_2) / len(mat_2)
   print('Average amount of gold with quality 2:', round(average_2, 2))

   average_3 = sum(mat_3) / len(mat_3)
   print('Average amount of gold with quality 3:', round(average_3, 2))

   average_4 = sum(mat_4) / len(mat_4)
   print('Average amount of gold with quality 4:', round(average_4, 2))
   print()
```

```
Average amount of gold with quality 1: 490.88
Average amount of gold with quality 2: 561.22
Average amount of gold with quality 3: 539.6
Average amount of gold with quality 4: 394.62
```

```
def const_cal():
   mat = []
   mat_1 = []
   mat_2 = []
   l = 0
   c = 0

   matrix_a, line, column = get_aur(mat, l, c)
   matrix_q = get_calitate(mat)

   for i in range(int(line)):
        for j in range(int(column)):
             if matrix_q[i][j] == 1:
             mat_1.append(matrix_a[i][j])
             if matrix_q[i][j] == 2:
             mat_2.append(matrix_a[i][j])

   with open('ConstCal.txt', 'w') as file:
        for i in mat_1:
             file.write(str(i) + ' ')
        file.write('\n')
        for i in mat_2:
             file.write(str(i) + ' ')

   print(Color('{green}Information was successfully written!{/green}\n'))
```

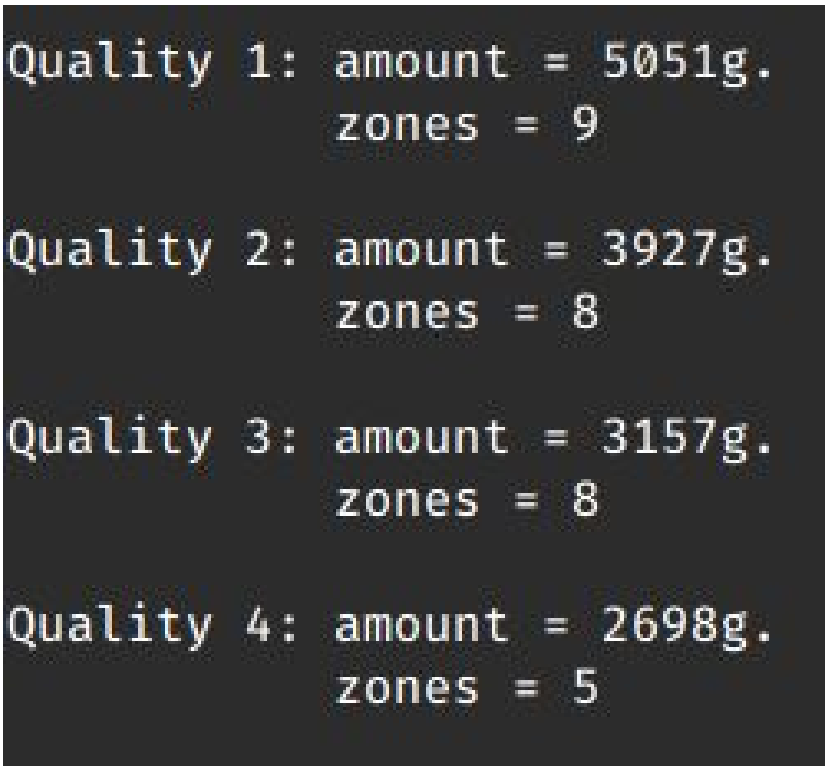
Библиография

- http://espressocode.top/find-average-list-python/

День 7

```
def QuickSort(A):
   if len(A) <= 1:
        return A
   else:
        q = random.choice(A)
        L, R, M = ([] for i in range(3))
        for elem in A:
             if elem > q:
             L.append(elem)
             elif elem < q:
             R.append(elem)
             else:
             M.append(elem)
        return QuickSort(L) + M + QuickSort(R)



def gold_sort():
   matrix_q, amount, mat, mat_1, mat_2, mat_3, mat_4 = ([] for i in range(7))
   l = 0
   c = 0

   matrix_a, line, column = get_aur(mat, l, c)
   matrix_q = get_calitate(mat)

   for i in range(int(line)):
        for j in range(int(column)):
             if matrix_q[i][j] == 1:
             mat_1.append(matrix_a[i][j])
             if matrix_q[i][j] == 2:
             mat_2.append(matrix_a[i][j])
             if matrix_q[i][j] == 3:
             mat_3.append(matrix_a[i][j])
             if matrix_q[i][j] == 4:
             mat_4.append(matrix_a[i][j])

   mat = [[len(mat_1), sum(mat_1)], [len(mat_2), sum(mat_2)], [len(mat_3),
sum(mat_3)], [len(mat_4), sum(mat_4)]]
```

```
mat = QuickSort(mat)
c = 1
for i,s in mat:
     print('Quality {}: amount = {}g.\n\t   zones = {}\n'.format(c, s, i))
     c+=1
```



Библиография

-