Задание 1.

Исходные данные:

Реализовать класс **Matrix** (матрица). Обеспечить перегрузку конструктора класса (метод

__init__()), который должен принимать данные (список списков) для формирования матрицы.
Подсказка: матрица — система некоторых математических величин, расположенных в виде прямоугольной схемы.
Примеры матриц: 3 на 2, 3 на 3, 2 на 4.

31
22
37
43
51
86

3
5
32
2
4
6
-1
64
-8

3
5
8
3
8
3
7
1

Следующий шаг — реализовать перегрузку метода __str__() для вывода матрицы в привычном виде.
Далее реализовать перегрузку метода __add__() для реализации операции сложения двух объектов класса Matrix (двух матриц). Результатом сложения должна быть новая матрица.
Подсказка: сложение элементов матриц выполнять поэлементно — первый элемент первой строки первой матрицы складываем с первым элементом первой строки второй матрицы и т. д.
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Matrix:
...     def __init__(self, a, b):
...         self.a = a
...         self.b = b
...
>>> def __str__(self):
...         return 'Matrix (%d, %d)' % (self.a, self.b)
...
>>> def __add__(self,other):
...         return Matrix(self.a + other.b, self.a + other.b, self.a + other.b, self.a + other.b)
...
>>> v1 = [55, 77, 88, 99]
>>> v2 = [55, 99, 88, 77]
>>> print(v1)
[55, 77, 88, 99]
>>> print(v2)
[55, 99, 88, 77]
>>> import numpy as np
>>> c = np.array([55, 77, 88, 99])
>>> d = np.array([55, 99, 88, 77])
>>> f = c + d
>>> print(f)
[110 176 176 176]
>>>
```

Задание 2.
Исходные данные:
Реализовать проект расчета суммарного расхода ткани на производство одежды. Основная сущность (класс) этого проекта — одежда, которая может иметь определенное название. К типам одежды в этом проекте относятся пальто и костюм. У этих типов одежды существуют параметры: размер (для пальто) и рост (для костюма). Это могут быть обычные числа: V и H, соответственно.
Для определения расхода ткани по каждому типу одежды использовать формулы: для пальто (V/6.5 + 0.5), для костюма (2*H + 0.3). Проверить работу этих методов на реальных данных.
Реализовать общий подсчет расхода ткани. Проверить на практике полученные на этом уроке знания: реализовать абстрактные классы для основных классов проекта, проверить на практике работу декоратора @property.
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Clothes:
...     def __init__(self, name, params1, params2):
...         self.name = name
...         self.params1 =params2 = params2
...
>>> class Coat(Clothes):
...     def __init__(self, size):
...         self.size = size
```

```
...         super().__init__()
...
>>> V = 48
>>> def my_size(V):
...     try:
...         a = int(input("Введите Ваш размер: "))
...     except zerodivisionerror:
...         return
...     s = V / 6.5 + 0.5
...     return s
...
>>> print(my_size(V))
Введите Ваш размер: 48
7.884615384615385
>>>
>>> class Costume(Clothes):
...     def __init__(self, height):
...         self.height = height
...         super().__init__()
...
>>> H = 180
>>> def my_height(H):
...     try:
...         b = int(input("Введите Ваш рост: "))
...     except zerodivisionerror:
...         return
...     s = 2 * H + 0.3
...     return s
...
>>> print(my_height(H))
Введите Ваш рост: 190
360.3
>>>
>>> @property
... def my_params(self):
...         return self.__params1, params2
...
>>> @my_params.setter
... def my_params(self, params1, params2):
...     if params1<40 and params2<150:
...         self.__params1, params2 = 40 and 150
...     elif params1>64 and params2>250:
...         self.__params1, params2 = 64 and 250
...     else:
...         self.__params1, params2 = params1, params2
...
>>> def get_Clothes_params1_params2(self):
...     return f"Расход ткани на пошив пальто и костюма {my_size(V) + my_height(H)}"
...
>>> print(f"Расход ткани на пошив пальто и костюма {my_size(V) + my_height(H)}")
Введите Ваш размер: 48
Введите Ваш рост: 190
```

Расход ткани на пошив пальто и костюма 368.18461538461537
>>>

Задание 3.
Исходные данные:
Реализовать программу работы с органическими клетками, состоящими из ячеек.
Необходимо создать класс Клетка. В его конструкторе инициализировать параметр, соответствующий количеству ячеек клетки (целое число). В классе должны быть реализованы методы перегрузки арифметических операторов: сложение (__add__()), вычитание (__sub__()), умножение (__mul__()), деление (__truediv__()). Данные методы должны применяться только к клеткам и выполнять увеличение, уменьшение, умножение и целочисленное (с округлением до целого) деление клеток, соответственно.
Сложение. Объединение двух клеток. При этом число ячеек общей клетки должно равняться сумме ячеек исходных двух клеток.
Вычитание. Участвуют две клетки. Операцию необходимо выполнять только если разность количества ячеек двух клеток больше нуля, иначе выводить соответствующее сообщение.
Умножение. Создается общая клетка из двух. Число ячеек общей клетки определяется как произведение количества ячеек этих двух клеток.
Деление. Создается общая клетка из двух. Число ячеек общей клетки определяется как целочисленное деление количества ячеек этих двух клеток.
В классе необходимо реализовать метод make_order(), принимающий экземпляр класса и количество ячеек в ряду. Данный метод позволяет организовать ячейки по рядам.
Метод должен возвращать строку вида *****\n*****\n*****..., где количество ячеек между \n равно переданному аргументу. Если ячеек на формирование ряда не хватает, то в последний ряд записываются все оставшиеся.
Например, количество ячеек клетки равняется 12, количество ячеек в ряду — 5. Тогда метод make_order() вернет строку: *****\n*****\n**.
Или, количество ячеек клетки равняется 15, количество ячеек в ряду — 5. Тогда метод make_order() вернет строку: *****\n*****\n*****.
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Cullale:
...     def __init__(self, number, width, height, sign):
...         self.number = number
...         self.width = int(width)
...         self.height = int(height)
...         self.sign = str(sign)
...
>>> def __str__(self):
...     rect = []
...     for i in range(self.height):
...         rect.append(self.sign * self.width)
...         rect = '*'.join(rect)
...     return rect
...
>>> b = Cullale(6, 6, 2, '*')
>>> c1 = [6, 6, 6, 6, 6, 6]
>>> c2 = [6, 6, 6, 6, 6, 6]
>>> def __add__(self,other):
```

```
...     return Cullale(self.c1 + other.c2, self.c1 + other.c2, self.c1 + other.c2, self.c1 + other.c2, self.c1
+ other.c2, self.c1 + other.c2)
...
>>> def __sub__(self, other):
...     return Cullale(self.c1 - other.c2, self.c1 - other.c2, self.c1 -other.c2, self.c1 - other.c2, self.c1 -
other.c2, self.c1 - other.c2)
...
>>> def __mul__(self,other):
...     return Cullale(self.c1 * other.c2, self.c1 * other.c2, self.c1 * other.c2, self.c1 * other.c2, self.c1
* other.c2)
...
>>> def __truediv__(self,other):
...     return Cullale(self.c1 / other.c2, self.c1 / other.c2, self.c1 / other.c2, self.c1 / other.c2, self.c1 /
other.c2, self.c1 / other.c2)
...
>>> import numpy as np
>>> a = np.array([25, 55, 77, 88, 55, 99])
>>> b = np.array([55, 77, 88, 25, 55, 99])
>>> c = a + b
>>> print(c)
[ 80 132 165 113 110 198]
>>>
>>> a = np.array([55, 88, 99, 55, 77, 88])
>>> d = np.array([55, 68, 98, 78, 88, 99])
>>> f = d * a
>>> print(f)
[3025 5984 9702 4290 6776 8712]
>>> g = np.array([55, 77, 88, 25, 45, 99])
>>> h = np.array([55, 77, 88, 25, 45, 99])
>>> z = g / h
>>> print(z)
[1. 1. 1. 1. 1. 1.]
>>>
>>> x = np.array([88, 99, 98, 89, 97, 85])
>>> y = np.array([15, 18, 25, 27, 32, 34])
>>> u = x - y
>>> print(u)
[73 81 73 62 65 51]
>>>
```