**ДОМАШНЕЕ ЗАДАНИЕ**

**Объектно-ориентированный подход**

**1. Создайте класс Tensor**

***Обязательные свойства:***

dimension: int, tuple – размерность (количество элементов, строк, столбцов и т.д. в зависимости от порядка)

data: list – значения

При обращении по имени к объекту класса Tensor должны выводиться значения в виде списка

**2. Создайте класс Matrix, наследник Tensor, для работы с матрицами второго порядка.**

***Обязательные свойства:***

dimension: tuple – размерность матрицы (количество строк, столбцов)

data: list – значения матрицы по строкам

**3. Напишите методы conv\_rc2i() и conv\_i2rc() для конвертации адресации матрицы (строка (r), столбец (c)) в индексы списка self.data (индекс (i)).**

**4. Перегрузите метод \_\_str\_\_().**

Матрица выводится по строкам. Определите, сколько знакомест нужно для вывода самого большого значения и выводите все значения столбцами с правым выравниванием. Интервал между столбцами – 2 символа. Межстрочный интервал – 1 символ. Матрица начинается со строки «[», оканчивается строкой «]».

**ПРИМЕР:**

[

0 1 2 3 4 5 6 7 8 9

10 473 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

40 41 42 43 44 45 46 47 48 49

50 51 52 53 54 55 56 57 58 59

60 61 62 63 64 65 66 67 68 69

70 71 72 73 74 75 76 77 78 79

80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99

]

**5. Перегрузите метод \_\_getitem\_\_() для получения доступа к элементам матрицы для чтения значений.**

При доступе к элементу по ключу [int, int] выводится значение.

В остальных случаях создаётся объект класса Matrix с соответствующими значениями.

Допустимые типы ключей: int, list, slice, tuple. Для ключа типа tuple для его элементов допускаются значения типов int, list, slice.

**ПРИМЕРЫ:**

**1. M**

[

0 1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

40 41 42 43 44 45 46 47 48 49

50 51 52 53 54 55 56 57 58 59

60 61 62 63 64 65 66 67 68 69

70 71 72 73 74 75 76 77 78 79

80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99

]

**2. M[1,1]**

11

**3. M[1]**

[

10 11 12 13 14 15 16 17 18 19

]

**4. M[-1]**

[

90 91 92 93 94 95 96 97 98 99

]

**5. M[1:4]**

[

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

]

**6. M[:4]**

[

0 1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

]

**7. M[4:]**

[

40 41 42 43 44 45 46 47 48 49

50 51 52 53 54 55 56 57 58 59

60 61 62 63 64 65 66 67 68 69

70 71 72 73 74 75 76 77 78 79

80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99

]

**8. M[:]**

[

0 1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

40 41 42 43 44 45 46 47 48 49

50 51 52 53 54 55 56 57 58 59

60 61 62 63 64 65 66 67 68 69

70 71 72 73 74 75 76 77 78 79

80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99

]

**9. M[1:7:2]**

[

10 11 12 13 14 15 16 17 18 19

30 31 32 33 34 35 36 37 38 39

50 51 52 53 54 55 56 57 58 59

]

**10. M[:, 1]**

[

1

11

21

31

41

51

61

71

81

91

]

**11. M[1:4, 1:4]**

[

11 12 13

21 22 23

31 32 33

]

**12. M[1:4, :4]**

[

10 11 12 13

20 21 22 23

30 31 32 33

]

**13. M[1:4, 4:]**

[

14 15 16 17 18 19

24 25 26 27 28 29

34 35 36 37 38 39

]

**14. M[1:4, :]**

[

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

]

**15. M[-1:]**

[

90 91 92 93 94 95 96 97 98 99

]

**16. M[-2::-2]**

[

80 81 82 83 84 85 86 87 88 89

60 61 62 63 64 65 66 67 68 69

40 41 42 43 44 45 46 47 48 49

20 21 22 23 24 25 26 27 28 29

0 1 2 3 4 5 6 7 8 9

]

**17. M[-2::-2,1:4]**

[

81 82 83

61 62 63

41 42 43

21 22 23

1 2 3

]

**18. M[:, :]**

[

0 1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39

40 41 42 43 44 45 46 47 48 49

50 51 52 53 54 55 56 57 58 59

60 61 62 63 64 65 66 67 68 69

70 71 72 73 74 75 76 77 78 79

80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99

]

**19. M[[1, 4]]**

[

10 11 12 13 14 15 16 17 18 19

40 41 42 43 44 45 46 47 48 49

]

**20. M[:, [1,4]]**

[

1 4

11 14

21 24

31 34

41 44

51 54

61 64

71 74

81 84

91 94

]

**21. M[[1, 4], [1, 4]]**

[

11 14

41 44

]

5.