МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной

техники и автоматизированных систем

Лабораторная работа №11

по дисциплине: ООП

тема: «**Знакомство с языком программирования Python. Базовые структуры данных.**»

Выполнил: студент группы ПВ-233
Мороз Роман Алексеевич

Проверили:
Морозов Данила Александрович

Белгород 2025

Цель работы: Познакомится с базовыми конструкциями языка. Получить навык создания простых приложений. Изучить базовые типы.

Вариант 2.

На вход подаются данные в форме двумерных «матриц», количество матриц заранее не определено, разделителем между матрицами являются строки. Для каждой матрицы найти все, которые удовлетворяют следующему условию: сумма элементов каждой строки совпадает с суммой элементов текущей. Форма матрицы может быть не полной. Формат вывода требуется соблюсти.

Пример матриц:

1 1 1 2
3 1 1 4
2 1 5 3

1 1 1 2
3 1 3 4
2 1 5 3

1 2 1 1
1 2 1 5
1 1 5 4

Результат:

Матрица 1:	1 1 1 2 →	1 2 1 1
	3 1 1 4	1 2 1 5
	2 1 5 3	1 1 5 4

```python
class Matrix:
    """
    Класс для представления матрицы и работы с суммами строк.

    Атрибуты:
        data (list of list of int): Данные матрицы, где каждый вложенный список представляет строку.

    Примеры:
        >>> m1 = Matrix([[1, 2], [3, 4]])
        >>> m2 = Matrix([[3, 4], [1, 2]])
        >>> m1 == m2
        True
        >>> m3 = Matrix([[1, 1], [2, 2]])
        >>> m1 == m3
        False
    """

    def __init__(self, data):
        self.__data = data

    def row_sums(self):
        """Возвращает список сумм строк матрицы."""
        return [sum(row) for row in self.__data]

    def __eq__(self, other):
        """Проверяет равенство матриц по суммам строк (порядок строк не учитывается)."""
        return sorted(self.row_sums()) == sorted(other.row_sums())

    def __str__(self):
        """Возвращает строковое представление матрицы."""
        return '\n'.join([' '.join(map(str, row)) for row in self.__data])


def process_matrices(input_data):
    """
    Обрабатывает входные данные с матрицами, разделенными пустыми строками,
    и возвращает пары совпадающих матриц.
```

```
    Аргументы:
        input_data (str): Входные данные в виде строки.

    Возвращает:
        str: Результат обработки или сообщение об ошибке.

    Примеры:
        >>> input_data = '1 2\\n3 4\\n\\n4 3\\n2 1'
        >>> print(process_matrices(input_data))
        Матрица на входе:
        1 2
        3 4
        <BLANKLINE>
        Матрица на выходе:
        4 3
        2 1
        <BLANKLINE>
        Матрица на входе:
        4 3
        2 1
        <BLANKLINE>
        Матрица на выходе:
        1 2
        3 4

        >>> input_data = '1 a\\n2 3'
        >>> print(process_matrices(input_data))
        Ошибка: неверный формат входных данных. Строки должны содержать только числа, разделенные пробелами.
    """
    matrices = []
    current_matrix_data = []

    for line in input_data.splitlines():
        line = line.strip()
        if line:
            try:
                row = list(map(int, line.split()))
```

```
                current_matrix_data.append(row)
            except ValueError:
                return "Ошибка: неверный формат входных данных. Строки должны содержать только числа, разделенные пробелами."
        else:
            if current_matrix_data:
                matrices.append(Matrix(current_matrix_data))
                current_matrix_data = []

    if current_matrix_data:
        matrices.append(Matrix(current_matrix_data))

    if not matrices:
        return "Нет матриц для обработки."

    output = []
    for i, matrix in enumerate(matrices):
        matching_matrices = []
        for j, other in enumerate(matrices):
            if i != j and matrix == other:
                matching_matrices.append(other)

        if matching_matrices:
            output.append(f"Матрица на входе:\n{matrix}\n\nМатрица на выходе:\n{matching_matrices[0]}")

    return '\n\n'.join(output) if output else "Нет совпадающих матриц."


# from io import StringIO
# input_data = StringIO("1 1 1 2\n3 1 1 4\n2 1 5 3\n\n1 1 1 2\n3 1 3 4\n2 1 5 3\n\n1 2 1 1\n1 2 1 5\n1 1 5 4")
# result = process_matrices(input_data.read())
# print(result)
```

```
import unittest
import doctest
```

```
from io import StringIO
from matrix_solver import Matrix, process_matrices

class TestMatrix(unittest.TestCase):
    def test_equality(self):
        m1 = Matrix([[1, 2], [3, 4]])
        m2 = Matrix([[3, 4], [1, 2]])
        self.assertEqual(m1, m2)

    def test_inequality(self):
        m1 = Matrix([[1, 2], [3, 4]])
        m3 = Matrix([[1, 1], [2, 2]])
        self.assertNotEqual(m1, m3)

class TestProcessMatrices(unittest.TestCase):
    def test_basic_case(self):
        input_data = StringIO("1 2\n3 4\n\n4 3\n2 1")
        result = process_matrices(input_data.read())
        expected = (
            "Матрица на входе:\n1 2\n3 4\n\nМатрица на выходе:\n4 3\n2 1\n\n"
            "Матрица на входе:\n4 3\n2 1\n\nМатрица на выходе:\n1 2\n3 4"
        )
        self.assertEqual(result, expected)

    def test_invalid_input(self):
        input_data = StringIO("1 a\n2 3")
        result = process_matrices(input_data.read())
        self.assertEqual(result, "Ошибка: неверный формат входных данных.
Строки должны содержать только числа, разделенные пробелами.")

    def test_empty_input(self):
        result = process_matrices("")
        self.assertEqual(result, "Нет матриц для обработки.")

    def test_task_matrix(self):
        input_data = StringIO("1 1 1 2\n3 1 1 4\n2 1 5 3\n\n1 1 1 2\n3 1 3 4\n2
1 5 3\n\n1 2 1 1\n1 2 1 5\n1 1 5 4")
        result = process_matrices(input_data.read())
```

```
        self.assertEqual(result, """Матрица на входе:
1 1 1 2
3 1 1 4
2 1 5 3

Матрица на выходе:
1 2 1 1
1 2 1 5
1 1 5 4

Матрица на входе:
1 2 1 1
1 2 1 5
1 1 5 4

Матрица на выходе:
1 1 1 2
3 1 1 4
2 1 5 3""")

if __name__ == '__main__':
    doctest.testmod(verbose=True)
    unittest.main()
```

```
❯ python3 test_matrix_solver.py
9 items had no tests:
    __main__
    __main__.TestMatrix
    __main__.TestMatrix.test_equality
    __main__.TestMatrix.test_inequality
    __main__.TestProcessMatrices
    __main__.TestProcessMatrices.test_basic_case
    __main__.TestProcessMatrices.test_empty_input
    __main__.TestProcessMatrices.test_invalid_input
    __main__.TestProcessMatrices.test_task_matrix
0 tests in 9 items.
0 passed and 0 failed.
Test passed.
......
----------------------------------------------------------------------
Ran 6 tests in 0.000s

OK
```

Вывод: Познакомились базовыми конструкциями языка. Получили навыки создания простых приложений. Изучили базовые типы.