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: Объектно-ориентированное программирование

Тема: Знакомство с языком программирования Python. Базовые структуры данных.

Выполнил: студент группы ПВ-223
Дмитриев А.А.
Проверил:
Черников С.В.

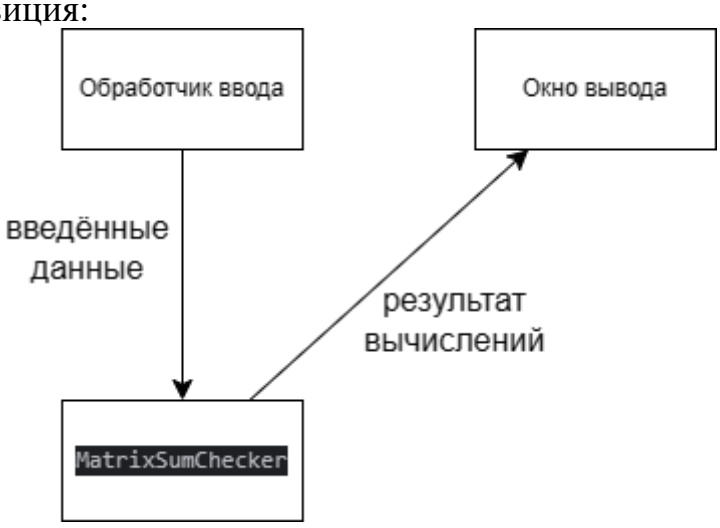
Белгород 2024 г.

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

Задание:
В соответствии с вариантом задания требуется выполнить объектную декомпозицию задачи. В качестве одного из обязательных объектов выделить «матрицу». Для реализации соблюдения условия задачи требуется использовать возможности перегрузки операторов. При выводе также требуется выполнить перегрузку соответствующего оператора.

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.

Объектная декомпозиция:



Код:

```
class matrix_sum_checker:
    __matrix = []

    def __init__(self, data):
        self.__matrix.clear()

        data = data.split("\n")
        buf = []
        for line in data:
            if not line.strip():
                self.__matrix.append(buf.copy())
                buf.clear()
            else:
                buf.append([int(x) for x in line.split()])

        if len(buf) > 0:
            self.__matrix.append(buf)

        del buf

    def find_matrices_with_equality_sums_by_rows(self):
        for i in range(len(self.__matrix)):
            for j in range(i + 1, len(self.__matrix)):
                if self.__is_sums_by_rows_equal(i, j):
                    yield (self.__matrix[i], self.__matrix[j])

    def __is_sums_by_rows_equal(self, m1_index, m2_index):
        if m1_index >= len(self.__matrix) or m2_index >= len(self.__matrix):
            raise IndexError

        if len(self.__matrix[m1_index]) != len(self.__matrix[m2_index]):
            return False

        amount_rows = len(self.__matrix[m1_index])
        for i in range(amount_rows):
            if sum(self.__matrix[m1_index][i]) != sum(self.__matrix[m2_index][i]):
                return False

        return True

    def find_matrices_with_equality_sums_by_rows_by_index(self, m_index):
        res = []
        for i in range(len(self.__matrix)):
            if i == m_index:
                continue

            if self.__is_sums_by_rows_equal(m_index, i):
                res.append(self.__matrix[i])
```

```
        return res

    def __getitem__(self, m_index):
        return self.find_matrices_with_equality_sums_by_rows_by_index(m_index)

    def __str__(self):
        return "".join([m1.__str__() + "\n--->\n" + m2.__str__() + "\n\n"
                    for m1, m2 in self.find_matrices_with_equality_sums_by_rows()])[:-2]


if __name__ == "__main__":
    data = \
        "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"

    res = matrix_sum_checker(data)
    print(res)
    print("-------------------------------------------------")
    print("by index 2:", res[2])
```

Пример работы программы:

```
C:\Users\dmitr\AppData\Local\Programs\Python\Python312\p
[[1, 1, 1, 2], [3, 1, 1, 4], [2, 1, 5, 3]]
--->
[[1, 2, 1, 1], [1, 2, 1, 5], [1, 1, 5, 4]]
-------------------------------------------------
by index 2: [[[1, 1, 1, 2], [3, 1, 1, 4], [2, 1, 5, 3]]]

Process finished with exit code 0
```

Вывод: В ходе лабораторной работы познакомились с базовыми конструкциями языка. Получили навыки создания простых приложений. Изучили базовые типы.