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.

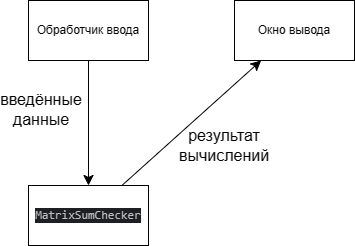
**Задание:**

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.

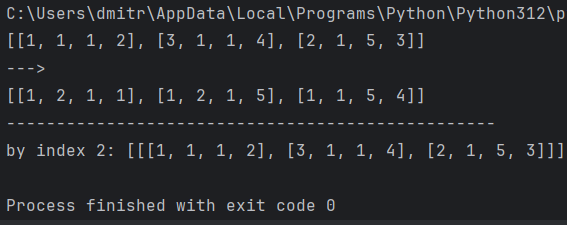
Объектная декомпозиция:



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:



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