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 №12
по дисциплине: ООП
тема: «**Знакомство с Python. Основные структуры данных**.»

Выполнил: студент группы ВТ-231
Масленников Д. А.
Проверили:
Буханов Д. Г.

Белгород 2025

Цель работы: приобретение практических навыков создания приложений на языке Python

Задание: Для выполнения лабораторной работы требуется установить интерпретатор Python версии 3.5+. Выполнить написание программы-сценария в соответствии с вариантом задания. Оформить отчет.

Вариант 14

Дано вещественное число k. Вставить между некоторыми цифрами, записанными в файле знак (+), (-), (*), (/), причем если возможно операция делания и умножения обязательна, так, чтобы значением получившегося выражения было число k (вычисление выполняется последовательно без учета приоритета операций). Например, в файле хранится: 1,2,3,4,2,2,1 если k=45, то подойдет следующая расстановка: 12+34*2/2-1.

Код программы:

```
class DigitExpressionBuilder:
    def __init__(self, digits, target):
        self.digits = digits
        self.target = target
        self.operators = ['+', '-', '*', '/']
        self.solutions = []

    def find_expressions(self):
        self._backtrack(0, "", 0)
        return self.solutions[0] if self.solutions else "решений не найдено"

    def _backtrack(self, index, current_expr, current_value):
        if index == len(self.digits):
            if current_value == self.target and (current_expr.count('*') > 0 or current_expr.count('/') >
0):
                self.solutions.append(current_expr)
            return

        for i in range(index, len(self.digits)):

            num_str = self.digits[index:i+1]
            num = int(num_str)

            if index == 0:
                self._backtrack(i+1, num_str, num)
            else:
                for op in self.operators:
                    new_value = current_value
                    if op == '+':
                        new_value += num
                    elif op == '-':
                        new_value -= num
                    elif op == '*':
                        new_value *= num
                    elif op == '/':
                        new_value /= num

                    self._backtrack(i+1, f"{current_expr}{op}{num_str}", new_value)
```

```
class DigitExpressionSolver:
    @staticmethod
    def solve_from_file(filename, target):
        with open(filename, 'r') as f:
            content = f.read().strip()
            digits = content.replace(',', '')

            if not digits.isdigit():
                return "ошибка: файл должен содержать только цифры и запятые"

            builder = DigitExpressionBuilder(digits, target)
            return builder.find_expressions()


if __name__ == "__main__":
    result = DigitExpressionSolver.solve_from_file("digits.txt",
45)
    print("результат:", result)
```

Вывод программы: