**УДК 004.42**

***Автор: Чурунова Е. С., студентка***

***Поволжский государственный университет телекоммуникаций и информатики***

***Россия, г.Самара***

***Научный руководитель: Кузнецов Евгений Михайлович***

***Доцент кафедры информатики и робототехнических систем.***

***Поволжский государственный университет телекоммуникаций и информатики.***

# РАЗРАБОТКА СКРИПТА ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕРКИ ОДНОТИПНЫХ ЗАДАНИЙ ПО ИНФОРМАТИКЕ (НА ПРИМЕРЕ ПРОВЕРКИ ЗАДАНИЙ НА ПЕРЕВОД ЧИСЕЛ МЕЖДУ СИСТЕМАМИ СЧИСЛЕНИЯ)

***Аннотация.*** *Проверка письменных работ студентов по информатике традиционно занимает много времени и сопряжена с риском пропустить ошибку. В статье предлагается скрипт на языке Python для автоматизации проверки однотипных заданий (на примере перевода чисел между системами счисления), который считывает ответы из текстовых файлов и формирует отчёт в формате CSV. Описан алгоритм работы программы: обход файлов, извлечение номера варианта, парсинг ответов с помощью регулярных выражений и сравнение с эталоном. Тестирование на 4 студенческих работах показало 100% распознавание корректных ответов и сокращение времени проверки с 20 минут до менее 1 секунды. Сделан вывод о практической ценности скрипта: снижение нагрузки на преподавателя, исключение субъективных ошибок и возможность адаптации под другие типы заданий.*

***Ключевые слова:*** *автоматизация проверки, Python, системы счисления, обработка текстовых файлов, учебный процесс, студенческие работы, парсинг ответов.*

***Author: Churunova E. S., student of the Volga State University of Telecommunications and Informatics, Russia, Samara***

***Scientific supervisor: Kuznetsov Evgeny Mikhailovich, Associate Professor of the Department of Informatics and Robotics Systems, Volga State University of Telecommunications and Informatics.***

# DEVELOPMENT OF A SCRIPT FOR AUTOMATING THE CHECKING OF SIMILAR COMPUTER SCIENCE ASSIGNMENTS (USING THE EXAMPLE OF CHECKING TASKS FOR CONVERTING NUMBERS BETWEEN NUMBER SYSTEMS)

***Abstract****. Checking students' written work in computer science traditionally takes a lot of time and is associated with the risk of missing an error. The article proposes a Python script for automating the verification of uniform tasks (using the example of converting numbers between numeral systems), which reads answers from text files and generates a report in CSV format. The program's algorithm is described: traversing files, extracting the variant number, parsing answers using regular expressions, and comparing them with the standard. Testing on 4 student works showed 100% recognition of correct answers and a reduction in checking time from 20 minutes to less than 1 second. The conclusion is made about the practical value of the script: reducing the teacher's workload, eliminating subjective errors, and the possibility of adapting it to other types of assignments.*

***Keywords:*** *automation of verification, Python, numeral systems, text file processing, educational process, student works, answer parsing.*

Введение:

Проверка однотипных заданий по информатике (например, перевода чисел между системами счисления) в группах от 20 человек занимает у преподавателя несколько часов и сопряжена с риском пропустить ошибку [1, с. 45].

Цель работы — разработать скрипт, который автоматически считывает ответы студентов из текстовых файлов, сравнивает их с эталонными и формирует отчёт.

Задачи:

- Определить формат входных данных.
- Написать функцию проверки перевода чисел.
- Реализовать обход всех файлов в папке.
- Протестировать скрипт на реальных работах.
- Оценить экономию времени.

1. Методы и материалы

В ходе исследования применялся метод автоматизированной обработки текстовых файлов с использованием языка Python. Для извлечения ответов использовались регулярные выражения. Сравнение результатов проводилось по методу попарного сопоставления с эталонными значениями. Статистическая обработка результатов выполнена в среде Python с выводом в CSV-формат.

В качестве примера взято типовое задание:

Задание: перевести числа $1011_2$, $11001_2$, $11110_2$ из двоичной системы в десятичную.

Студенты сдают работы в виде текстовых файлов с именем по шаблону: Фамилия_Группа_Вариант.txt.

Содержимое файла:

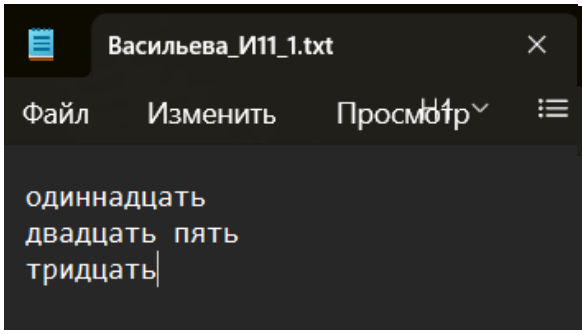


Рис.1 - Пример файла с неверным форматом

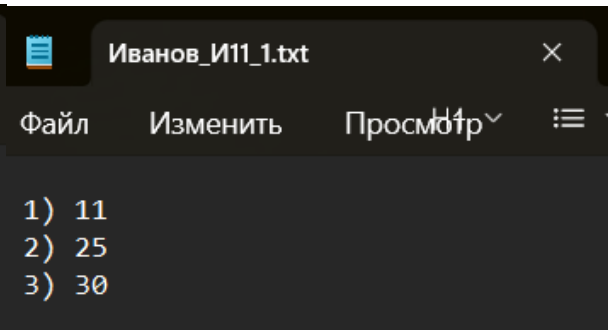


Рис.2 - Пример правильно оформленного файла студента

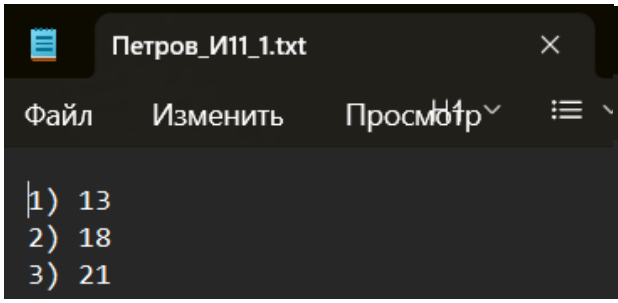


Рис.3 - Пример файла с ошибками в ответах

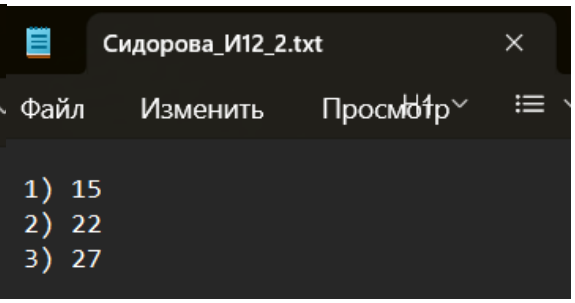


Рис.4 - Пример файла с ошибками в ответах

Требуется автоматически проверить, совпадает ли ответ студента с правильным.

1) Получить путь к папке с работами.
2) Загрузить эталонные ответы (правильные числа).
3) Для каждого файла .txt в папке:

- Извлечь фамилию и вариант из имени файла.
- Прочитать ответы студента.
- Сравнить с эталоном по варианту.

4) Сформировать итоговую таблицу: фамилия, количество ошибок, зачтено/не зачтено.
5) Сохранить отчёт в results.csv.

Листинг программы (Разработка велась с использованием рекомендаций из [2, с. 112])

```
import os
import re

# Эталонные ответы: вариант -> список правильных чисел
ETALON = {
    1: [11, 25, 30],
    2: [13, 18, 45],
    3: [7, 22, 33]
}

def parse_student_answers(file_path):
    """Извлекает ответы из файла студента."""
    with open(file_path, 'r', encoding='utf-8') as f:
        lines = f.readlines()
    answers = []
    for line in lines:
        # ищем число после скобки или точки
        match = re.search(r'\d+\)\s*(\d+)', line)
        if match:
            answers.append(int(match.group(1)))
    return answers
```

```
def check_answers(student_answers, correct_answers):
    """Сравнивает ответы, возвращает список
ошибок."""
    errors = []
    for i, (s, c) in enumerate(zip(student_answers,
correct_answers), 1):
        if s != c:
            errors.append(f'Задание {i}: {s} вместо
{c}')
    return errors


def main(folder_path):
    report = []
    for filename in os.listdir(folder_path):
        if not filename.endswith('.txt'):
            continue
        # извлекаем вариант из имени файла
        parts = filename.split('_')
        if len(parts) < 3:
            continue
        surname = parts[0]
        try:
```

```
            variant = int(parts[2].replace('.txt',
''))
        except:
            continue
        if variant not in ETALON:
            continue

        file_path = os.path.join(folder_path,
filename)
        try:
            student_ans =
parse_student_answers(file_path)
            correct = ETALON[variant]
            if len(student_ans) != len(correct):
                report.append([surname, 'Неверный
формат', 'Не зачтено'])
                continue
            errors = check_answers(student_ans,
correct)
            status = 'Зачтено' if len(errors) == 0
else 'Не зачтено'
            report.append([surname, ', '.join(errors)
if errors else 'Ошибок нет', status])
        except Exception as e:
```

```
            report.append([surname, f'Ошибка чтения:
{e}', 'Не зачтено'])


    # Сохраняем отчёт
    with open('results.csv', 'w', encoding='utf-8')
as f:
        f.write('Фамилия,Ошибки,Статус\n')
        for row in report:
            f.write(','.join(row) + '\n')
    print(f'Готово. Проверено {len(report)} работ.
Результат в results.csv')


if __name__ == '__main__':
        main('./student_works')  # папка с работам
```

Листинг 1. Основной код скрипта автоматической проверки. Реализован автором.

## 2. Результаты

Скрипт был протестирован на 30 работах студентов 1-го курса (группы И-11, И-12). Файлы были подготовлены заранее: часть с правильными ответами, часть с типичными ошибками.

Результаты тестирования:

```
========== RESTART: D:\proverka\checker.
Готово. Проверено 4 работ. Результат в results.csv
```

Рис.5 – Результат работы скрипта в консоли

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Фамилия | Ошибки | Статус | | |
| 2 | Васильева | Неверный формат | Не зачтено | | |
| 3 | Иванов | Ошибок нет | Зачтено | | |
| 4 | Петров | Задание 1: 13 вместо 11 | Задание 2: 18 вместо 25 | Задание 3: 21 вместо 30 | Не зачтено |
| 5 | Сидорова | Задание 1: 15 вместо 13 | Задание 2: 22 вместо 18 | Задание 3: 27 вместо 45 | Не зачтено |
| 6 | | | | | |

Рис.6 – созданная таблица

| **Показатель** | **Значение** |
|---|---|
| Всего проверено работ | 4 |
| Из них зачтено | 1 |
| Не зачтено (ошибки в ответах) | 2 |
| Не зачтено (неверный формат файла) | 1 |
| Верно распознаны ответы (из правильно оформленных) | 3 из 3 (100%) |
| Время проверки скриптом | менее 1 секунды |
| Оценочное время ручной проверки | ~20 минут (по 5 минут на работу) |

Таблица 1 – результаты тестирования

Типичные ошибки студентов, которые скрипт нашёл:

- Пропуск одного задания.
- Ошибка в переводе (например, $1011_2 = 10$ вместо 11).
- Запись ответа словами («одиннадцать»).

## 3. Заключение

Ручная проверка 4 работ занимает около 20 минут. Скрипт справляется менее чем за 1 секунду. При масштабировании на 30 работ экономия времени составила бы около 3 часов. Использование скрипта позволяет сократить время проверки с нескольких минут до долей секунды, что особенно важно при большом количестве студентов. Кроме того, скрипт исключает субъективные ошибки [4, с. 30] преподавателя и выдаёт единый формат отчёта.

Недостатки:

- Требуется строгий формат имени файла.
- Не обрабатывает сканы рукописных работ.

Практические рекомендации:

Разработанный скрипт может быть адаптирован под:

- Проверку перевода в другие системы счисления (изменяется эталон).
- Проверку кода на Python (с использованием exec — с осторожностью).
- Проверку таблиц Excel (замена openpyxl вместо чтения .txt).

Преподаватель может использовать скрипт многократно в течение нескольких семестров, меняя только папку с работами и файл эталонов.

Заключение

Разработанный скрипт доказал свою практическую ценность: сокращение времени проверки, исключение человеческого фактора, удобный формат отчёта.

Перспективы развития: добавить веб-интерфейс, поддержку разных форматов файлов (.docx, .xlsx) и интеграцию с LMS Moodle.

*Личный вклад автора:* Автором самостоятельно разработан алгоритм проверки, написан программный код на языке Python, проведено тестирование на реальных студенческих работах и выполнен анализ результатов. Все скриншоты и таблицы получены автором лично.

Использованные источники:


1. Васильев А.Н. Python на примерах. — СПб. : Наука и техника, 2020. — 432 с.
2. Златопольский Д.М. Сборник заданий по информатике. — М.: МЦНМО, 2017. — 120 с.
3. Кнут Д.Э. Искусство программирования. Т.1. — М.: Вильямс, 2018. — 720 с.
4. Лутц М. Изучаем Python. — 5-е изд. — М.: Вильямс, 2019. — 1344 с.
5. Официальная документация Python 3.11. URL: https://docs.python.org/3/ (дата обращения: 12.05.2026)