Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий и систем связи

Лабораторная работа №2

Вариант №1

Выполнил(и:)

Гусев Я.А.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

Введение

В данной работе я реализую алгоритм, который будет угадывать студента по его характеристикам, а также создам граф этого алгоритма.

Задание 1

Подготовка

Создадим словарь students_char (Рисунок 1), содержащий в себе характеристики всех одногруппников. В качестве ключа словаря выступает имя студента, а в качестве значения список характеристик типа list.

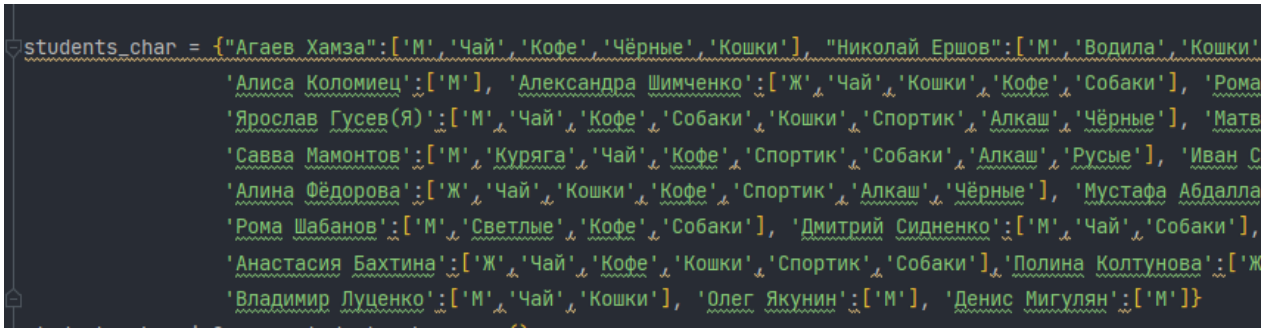

Рисунок 1 – Словарь со студентами

Далее создадим список tags всех существующих характеристик (тегов) (Рисунок 2) и список вопросов, которые наша программа будет задавать (Рисунок 3).

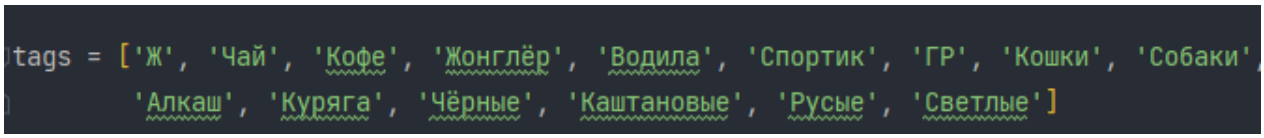

Рисунок 2 – Список характеристик

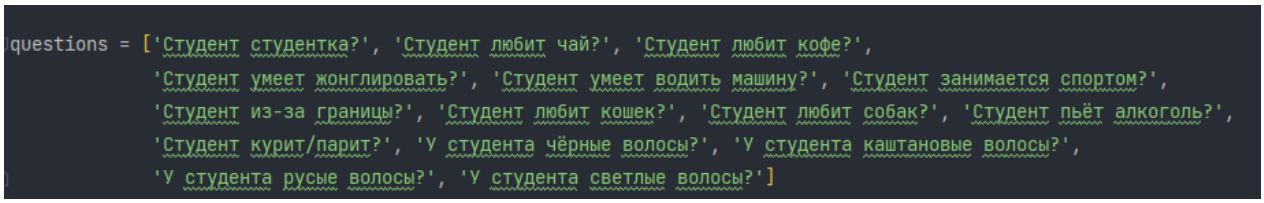

Рисунок 3 – Список вопросов

Для дальнейшего удобства вопросы располагаем в том же порядке, что и соответствующие им теги.

Создадим копии наших списков для реиграбельности (Рисунок 4) и (Рисунок 5).

```
tags_save = tags.copy()
```

Рисунок 4- Копия тегов

```
questions_inGame = questions.copy()
```

Рисунок 5 – Копия вопросов

Создание алгоритма

Для начала создадим цикл While True, в котором будет проходить вся игра. Сгенерируем случайный вопрос и зададим его пользователю (Рисунок 6).

```
question_number = randint(0, len(questions_inGame) - 1)
question = questions_inGame[question_number]
tag = tags[question_number]
print(question)
answer = input('Д/Н/ХЗ: ')
```

Рисунок 6 – Генерация случайного вопроса

Для начала получим случайный номер вопроса с помощью функции randint библиотеки random. Далее, используя полученный номер как индекс вопроса и тега, получаем вопрос и тег из списков questions_inGame и tags соответственно. Задаём вопрос и получаем ответ в переменную answer.

Создаём пустой словарь (Рисунок 7), в который будем вносить подходящих нам студентов.

```
students_char_inGame_otbor = dict()
```

Рисунок 7 – Новый словарь

Ответ да

Далее действуем по ситуации. Если ответ – да (Рисунок 8), то проходимся по всем студентам в словаре и проверяем их значение на наличие соответствующего тега.

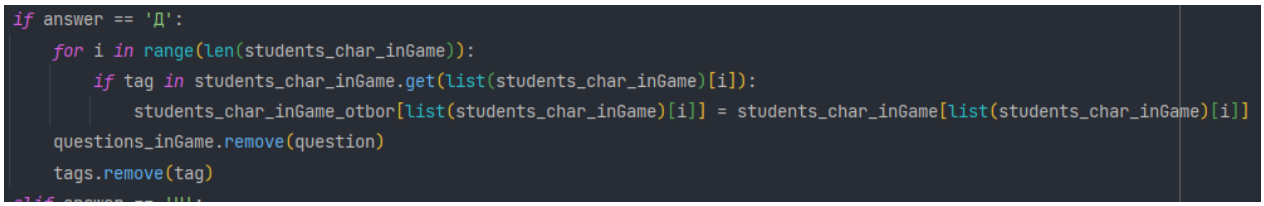

Рисунок 8 – Ответ да

Если тег есть, добавляем студента в наш новый словарь. Удаляем заданный вопрос и тег во избежание повтора.

Ответ нет

Если же ответ – нет (Рисунок 9), действуем похожим образом, но наоборот, если у студента нет тега, то он проходит в наш новый словарь.

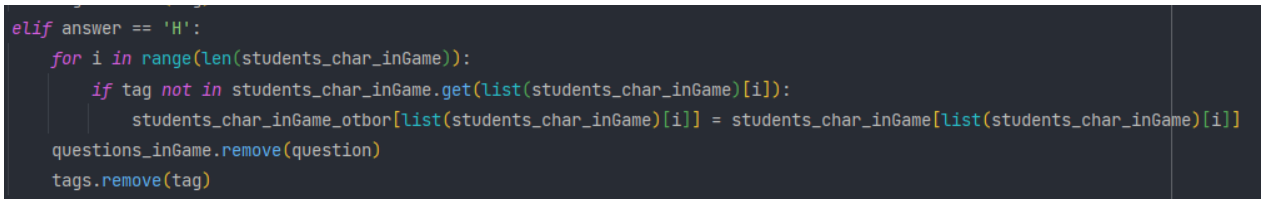

Рисунок 9 – Ответ нет

Ответ ХЗ

При ответе ХЗ (Рисунок 10) алгоритм начинает цикл заново.

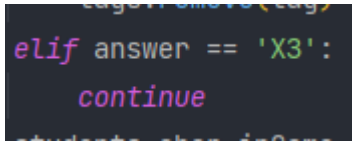

Рисунок 10 – Ответ ХЗ

В конце цикла сохраняем наш словарь с прошедшими студентами.

Также добавим в начале цикла while два условия, при которых игра заканчивается (Рисунок 11). Первое условие – в словаре не осталось студентов. Второе – в словаре остался один студент – загаданный пользователем.

```
if len(students_char_inGame) == 0:
    print('Я сдаюсь')


if len(students_char_inGame) == 1:
    print(f'Человек, которого вы загадали это: {list(students_char_inGame)[0]}')
```

Рисунок 11 – Конец игры

Реиграбельность

Для того, чтобы каждый раз не перезапускать игру, добавим в неё реиграбельность. Для начала создадим функцию restart() (Рисунок 12), которая в случае отрицательного ответа закончит игру, а в случае положительного – вернёт все словари и списки в изначальное состояние.

```
def restart():
    answer = input('Хотите сыграть ещё раз? (Д/Н): ')
    if answer == 'Д':
        global questions_inGame, students_char_inGame, tags
        questions_inGame = questions.copy()
        students_char_inGame = students_char.copy()
        tags = tags_save.copy()
    else:
        print('Конец игры')
        return 0
```

Рисунок 12 – Функция рестарта игры

Теперь вставим выполнение этой функции в конце игры (Рисунок 13).

```
if len(students_char_inGame) == 0:
    print('Я сдаюсь')
    if restart() == 0: break

if len(students_char_inGame) == 1:
    print(f'Человек, которого вы загадали это: {list(students_char_inGame)[0]}')
    if restart() == 0: break
```

Рисунок 13 – Рестарт

Если функция вернёт 0 (в случае если пользователь хочет прекратить игру), мы выходим из главного цикла при помощи break и игра прекращается.

Задание 2 – граф

Создадим граф реализованного нами алгоритма (Рисунок 14).

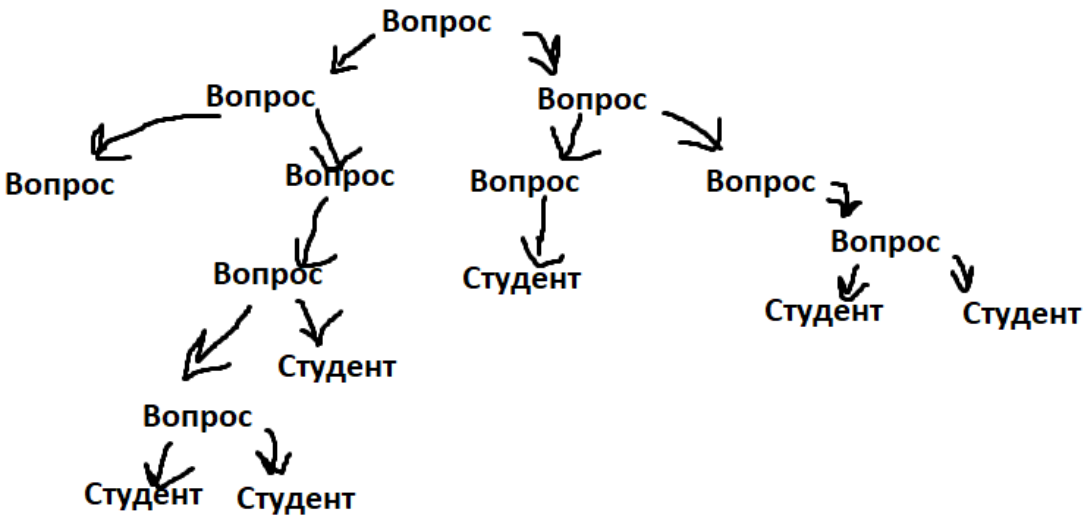

Рисунок 14 - Дерево

Вывод

Я сделал алгоритм по типа известной игры «Акинатор» на языке Python при помощи словарей. Также я составил граф созданного алгоритма.

Список литературы.