**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «МКиИТ»

**Отчет по летней практической работе**

На тему «Распознавание и классификация речи»

Выполнил: студент группы БВТ2105

Вотченников Дмитрий Александрович

Проверил: Жмакин Евгений

Москва

2022

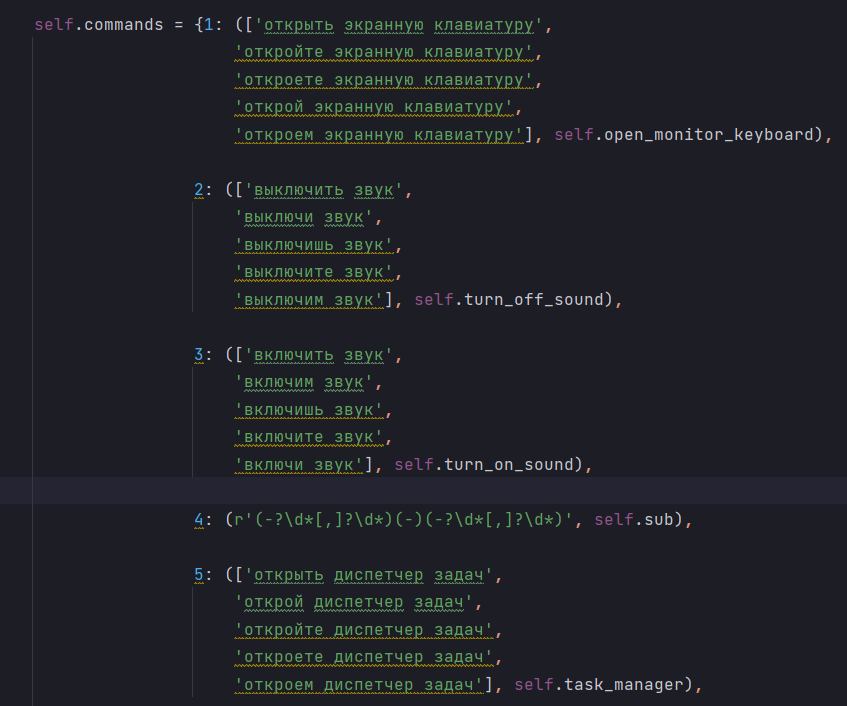
**Цели и задачи**

Целью работы является создать программу, способную распознавать определённую группу команд и выполнять их. При этом необходимо учесть все возможные формулировки и исключать недопустимые значения ф программе. По возможности создать небольшое приложение, загрузить полученные результаты на удаленный репозиторий: “GitHub”

**Вариант: 8**

**Ход выполнения**

На Рис.1-5 продемонстрирован код, в котором осуществляется проверка на наличие команды в фразе, а также показан метод(compare\_words), которая делает предположение команды, если она не была распознана сразу, все возможные случаи проверяются в методе “sending\_result”



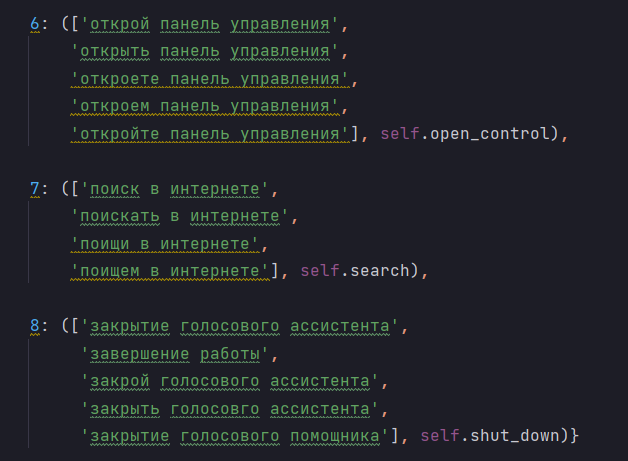


Рис.1 – Словарь, содержащий команды и методы связанные с ними

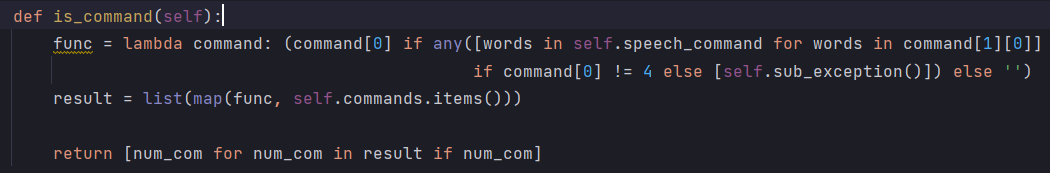


Рис.2 - Основной метод на проверку наличия команды в фразе

Здесь, переменная “speech\_command” является результатом произнесенной фразы, переменная “commands” является словарём, в котором хранятся все команды с их возможными вариациями, а также “привязанные” к ним методы класса (которые необходимы для их выполнения). Метод “ sub\_exception” обрабатывает случай, когда идёт проверка на наличие команды: “Вычитание двух чисел”

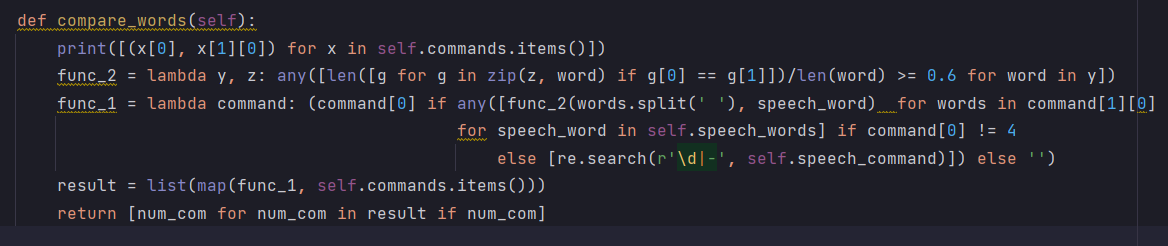


Рис.3 – Метод, предполагающий произнесенную команду

Сначала я создаю переменную “result”, которой я присваиваю результат выполнения функции “map”, “self.commands.items()” – это кортеж, первым элементом которого является номер команды, а вторым идёт список из возможных вариаций команды, а также метод класса, для выполнения команды. Всё это поэлементно отправляется в функцию “func\_1”, которая в свою очередь вызывает другую функцию “func\_2”. Сам алгоритм заключается в следующем: я беру слово из произнесенной фразы и беру одну вариацию команды, затем выбранное из фразы слово поэлементно сравнивается с каждым словом из команды, и если совпадение составило не менее 60%, то идёт запоминание этой команды как потенциально возможной. В конце идёт фильтрация значений, чтобы остались только номера команд (а не пустые строки).

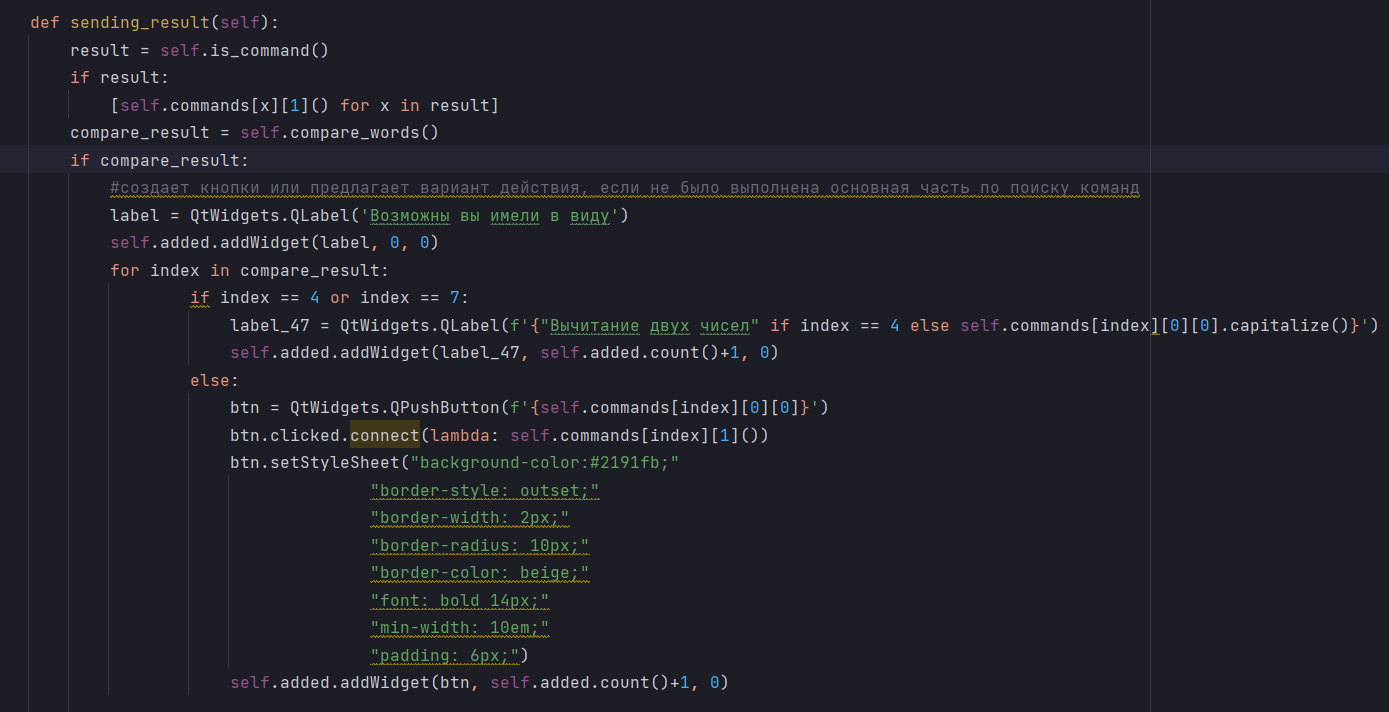


Рис.4 – Метод класса, осуществляющий вывод результатов в приложение

Этот метод работает следующим образом: сначала вызывается метод “ is\_command”, которая проверяет наличие команды в фразе. Если после её выполнения список не пустой, то идёт выполнение команды, а если нет, то идёт проверка на возможные команды через метод “compare\_words”. Если есть хоть одно совпадение с какой-нибудь командой, то идёт предложение пользователю либо выполнять команду, либо повторить выполнение команды.

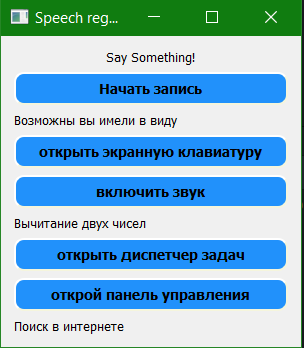
****

Рис.5 – Пример выполнения программы, при произношении фразы “-3 поиск в открыть включить”

На Рис.6-7 продемонстрирован код и выполнение первой команды: “ Открытие диспетчера задач”



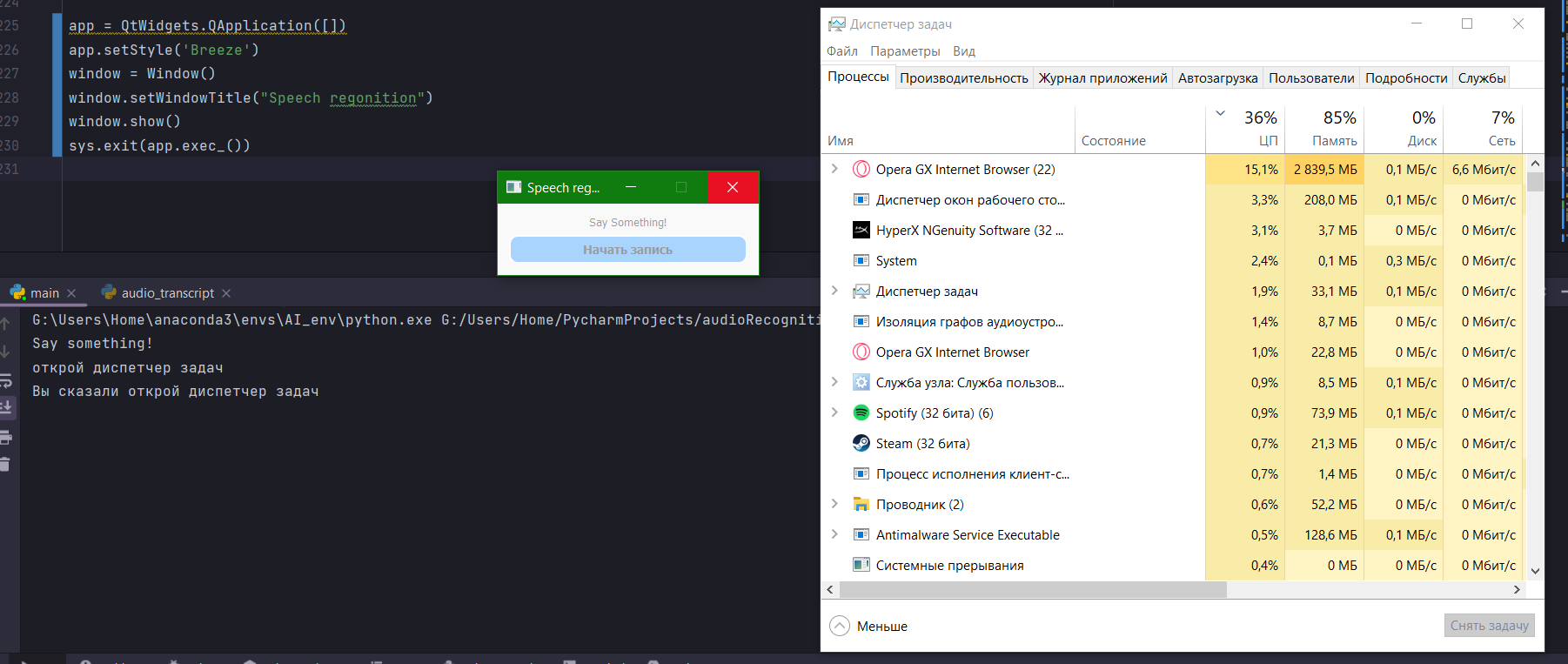


Рис.6 – Результат команды “открой диспетчер задач”

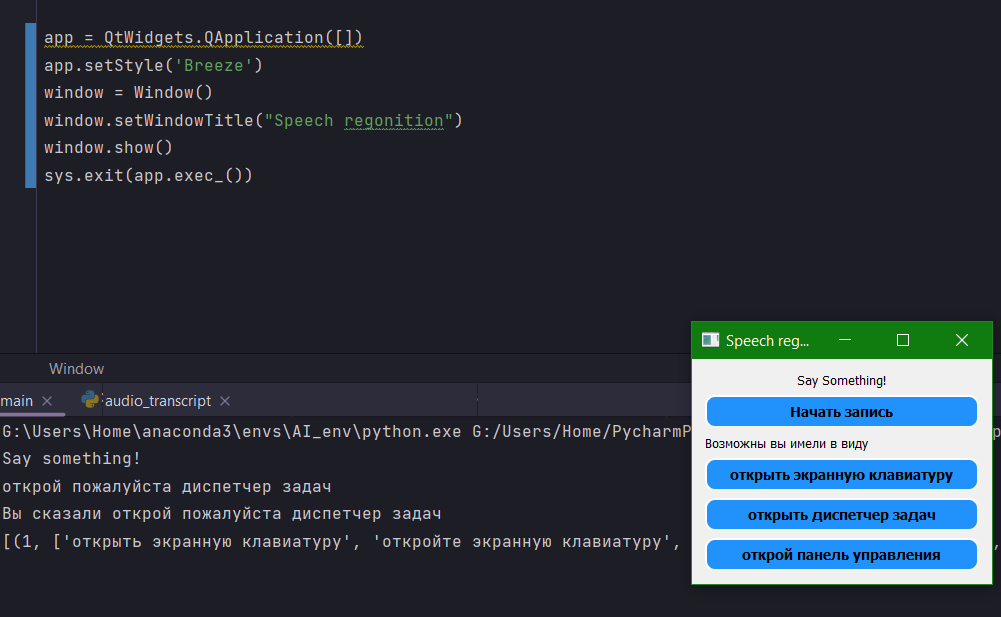
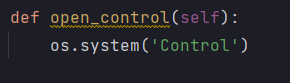


Рис.7 – Результат выполнения команды “открой пожалуйста диспетчер задач”

На Рис.8 – 9 показан результат выполнения команды “Открой панель управления” и команды “Открой пожалуйста панель управления”



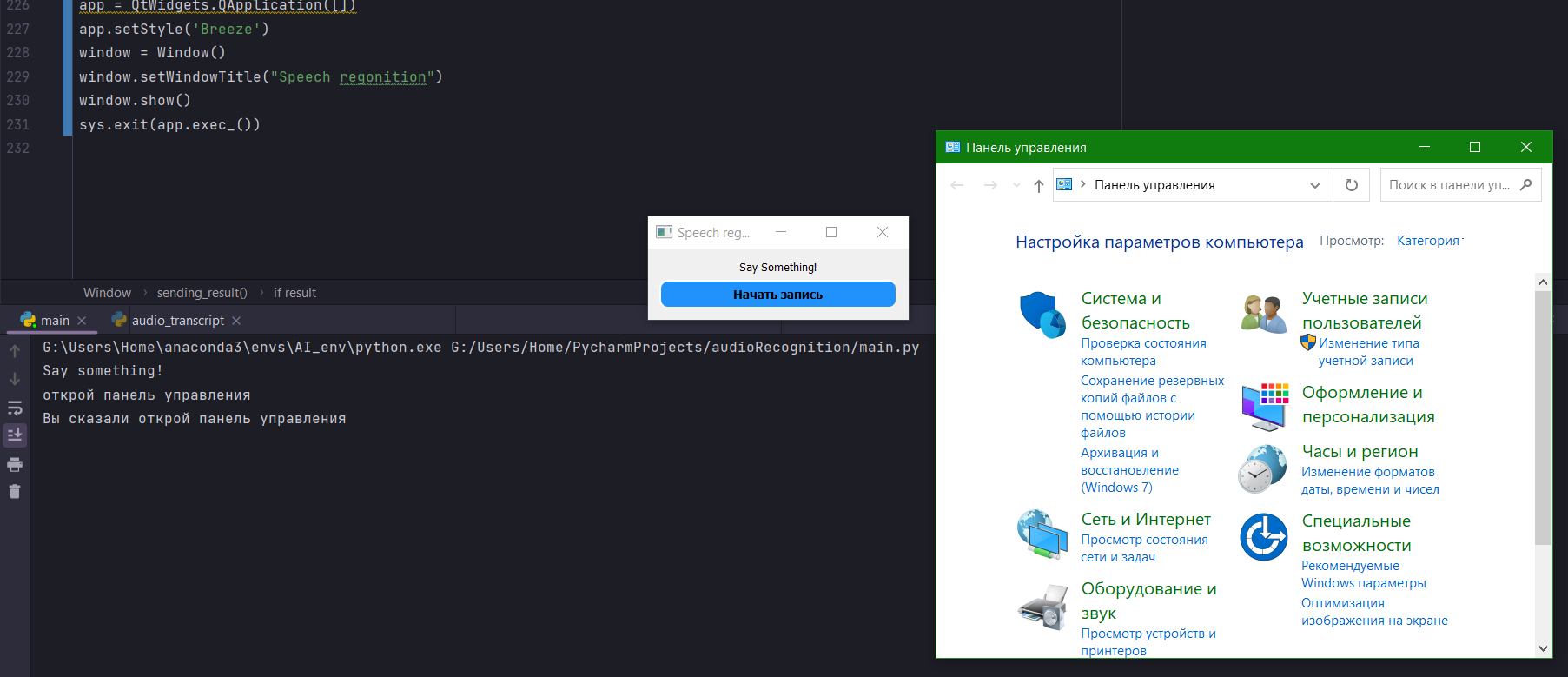


Рис.8 – Результат выполнения команды “открой панель управления”

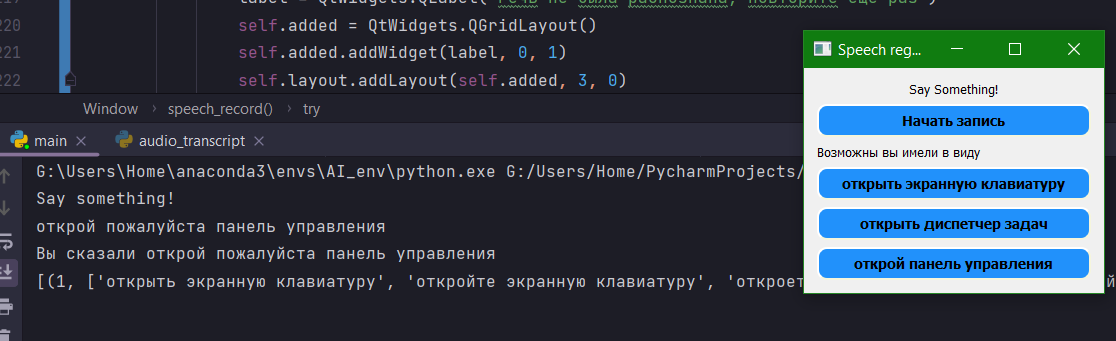


Рис.9 – Результат выполнения команды “Открой пожалуйста панель управления”

На Рис.10 – 11 продемонстрирована работа команд “Поиск в интернете Иван Грозный”, “Поиск в “, “Поиск в интернет Иван грозный и поищи в интернете реформы Ивана Грозного”.

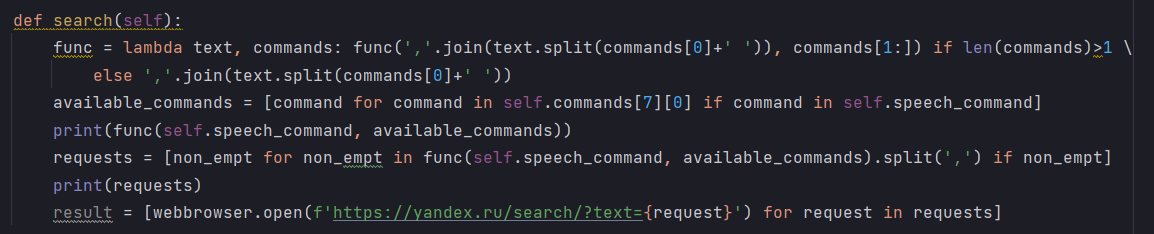
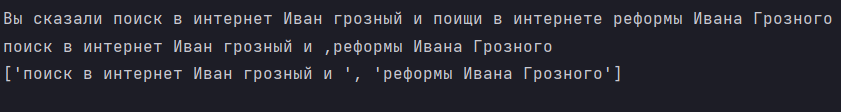
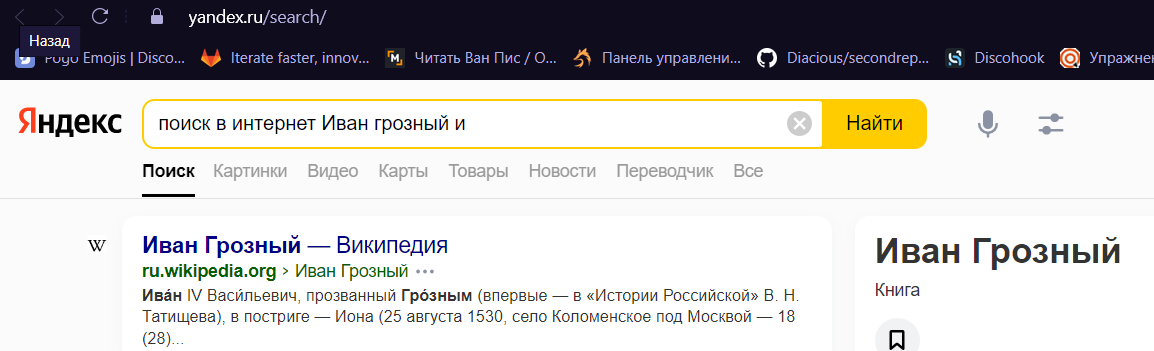


Рис.10 – Метод, отвечающий за обработки команды “поиск в интернете”

Метод “search” позволяет обрабатывать несколько поисковых запросов, например, если будет сказано: “Поиск в интернет Иван грозный и поищи в интернете реформы Ивана Грозного”, то будет выведено в 2 отдельных вкладках эти запросы, сначала “Иван Грозный”, а затем “Реформы Ивана Грозного”. Рис.11 демонстрирует данное поведение





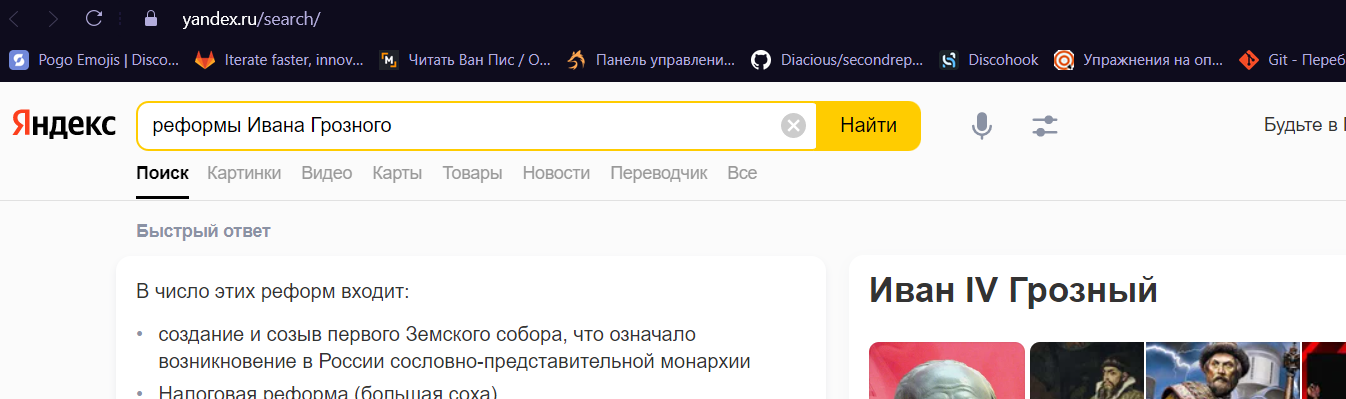
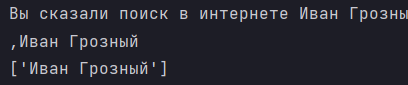


Рис.11 – Пример обработки нескольких запросов



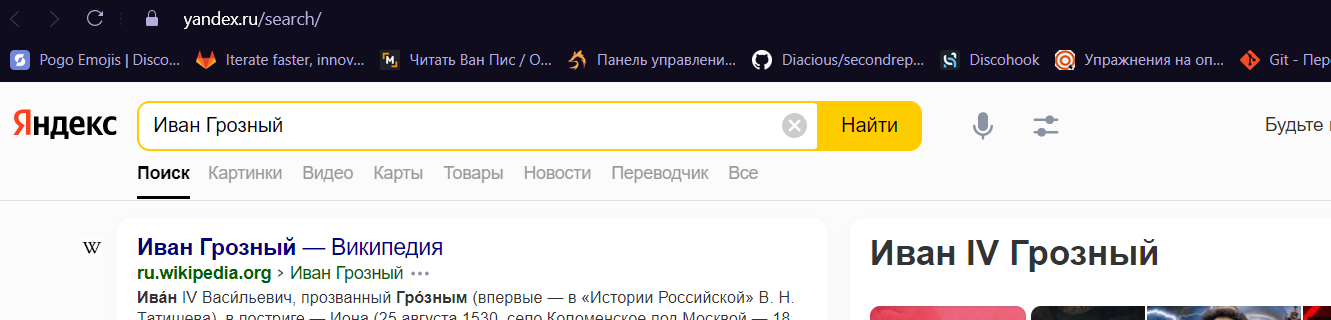


Рис.12 – Результат выполнения команды “поиск в интернете Иван Грозный”

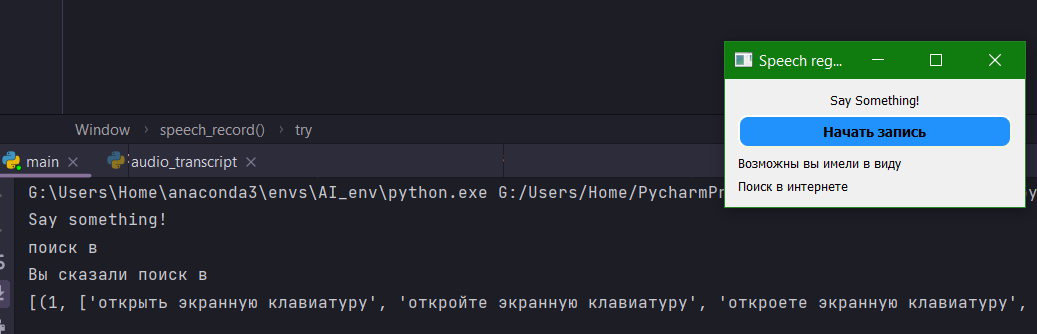
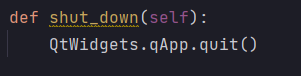


Рис.13 – Результат выполнения команды “поиск в”

На Рис.14 – 15 продемонстрирована работа команд “Закрой голосового помощника” и “Закрой пожалуйста голосового помощника”



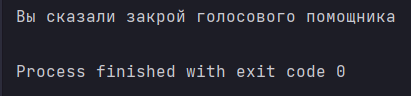


Рис.14 – Результат работы команды “закрой голосового помощника”

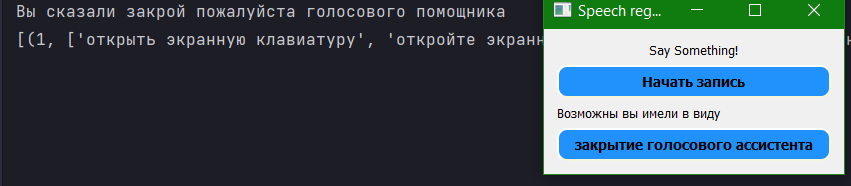


Рис.15 – Результат работы команды “закрой пожалуйста голосового помощника”

На Рис.16 – 17 продемонстрирована работа команд “открыть экранную клавиатуру” и “открой пожалуйста экранную клавиатуру”

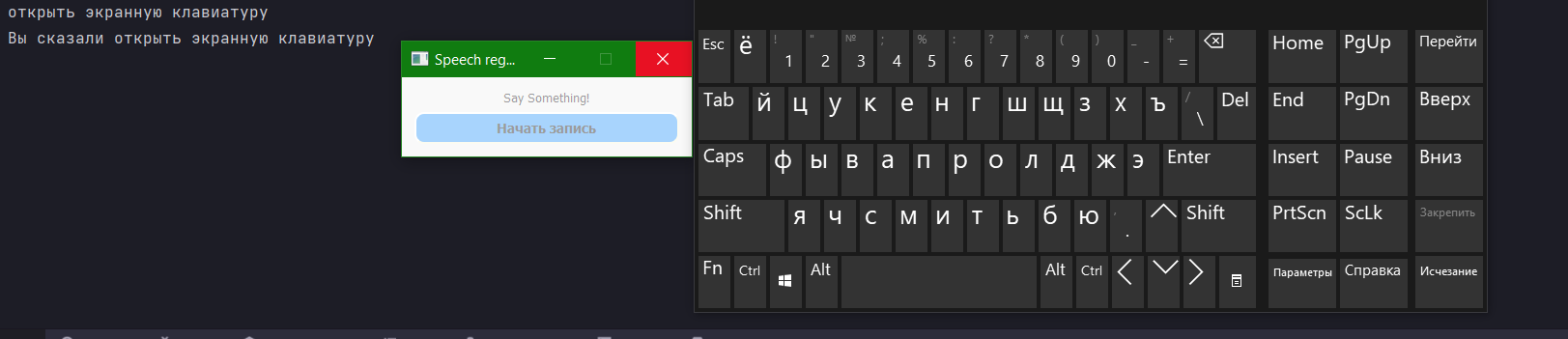


Рис.16 – Результат команды “открыть экранную клавиатуру”

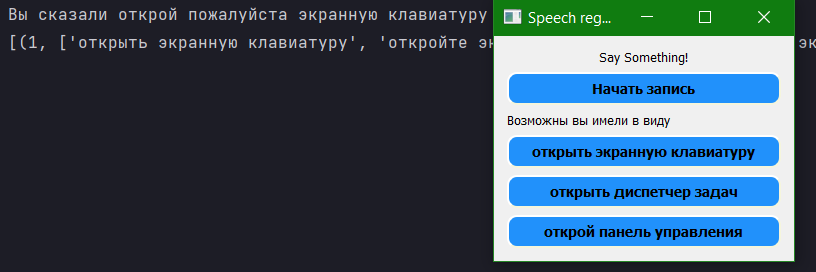
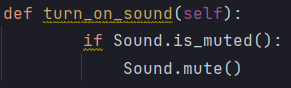


Рис.17 – Результат команды “открой пожалуйста экранную клавиатуру”

На Рис.18-19 показаны результаты выполнения команд “включи звук”, “выключи звук”.



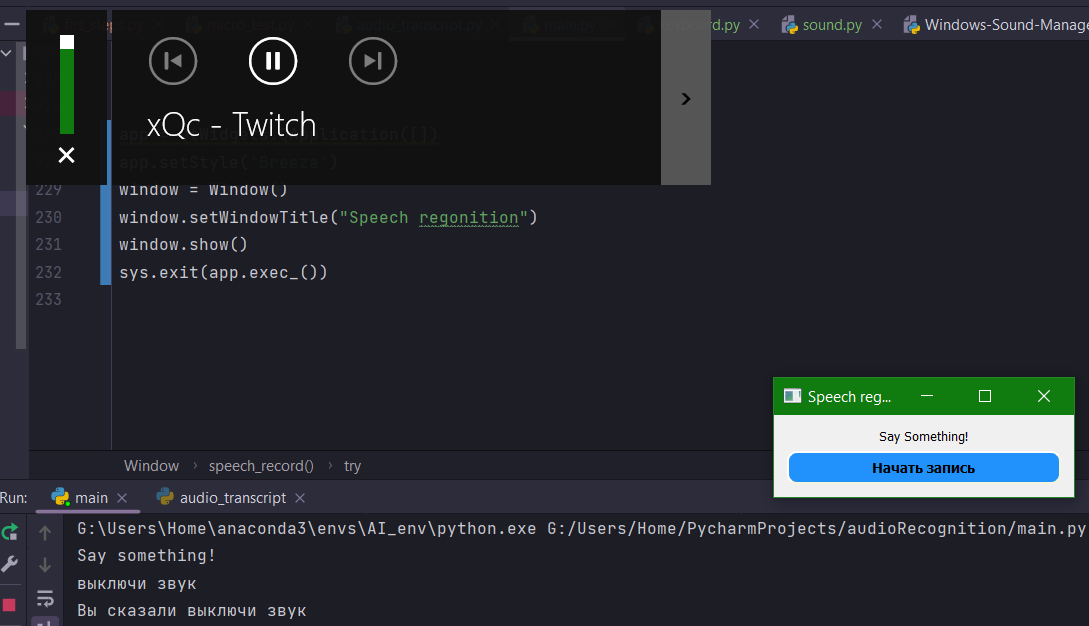


Рис.18 – Результат работы команды “выключи звук”

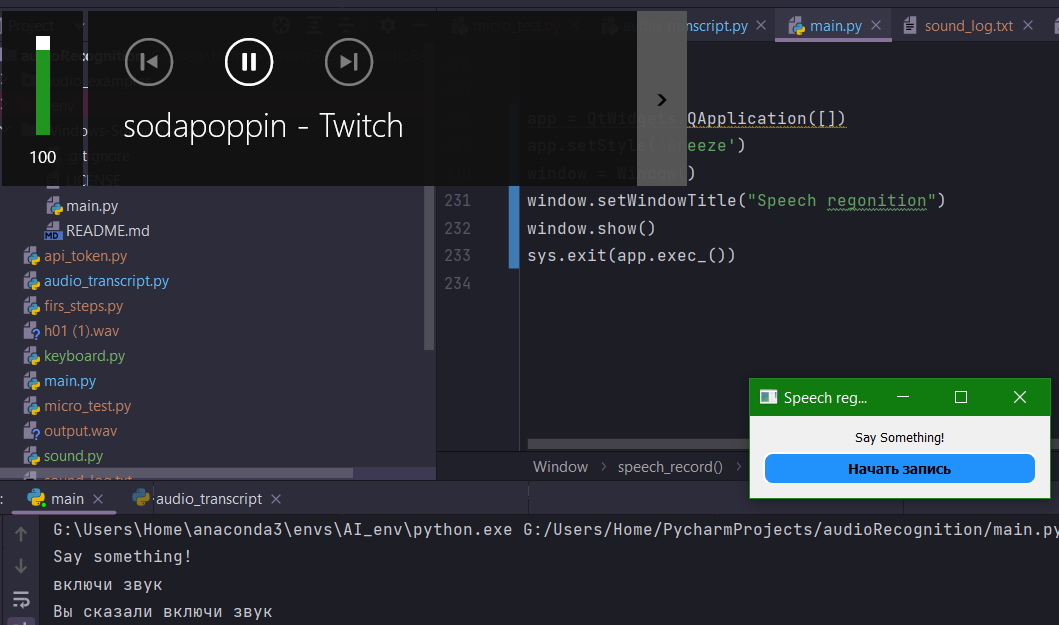


Рис.19 – Результат работы команды “включи звук”

На Рис.20 – 21 показаны результаты работы команд “-1,5 -10,5” и “-1 - - 2”.

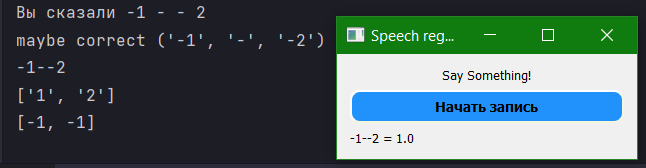
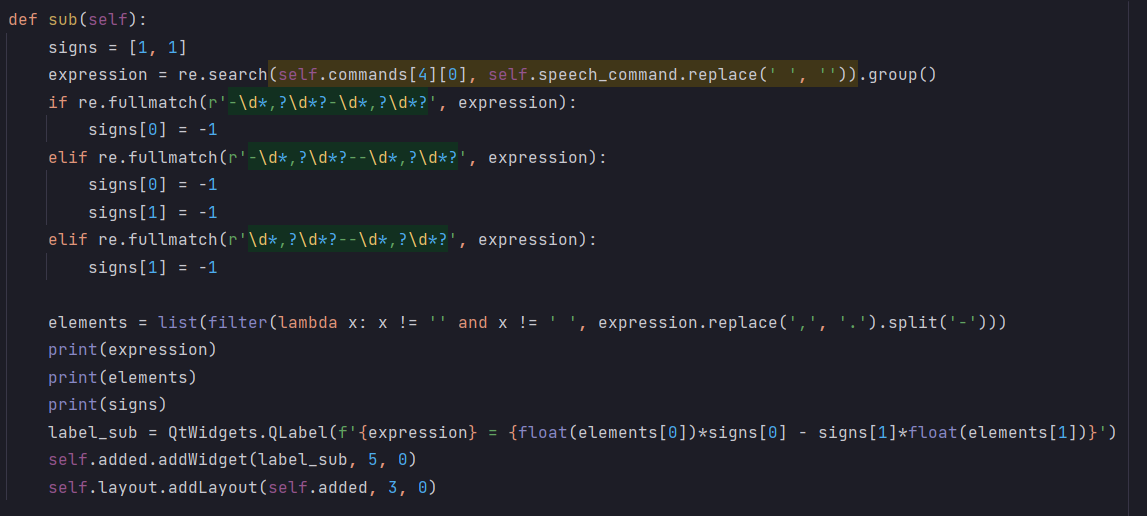


Рис.20 – Результат работы команды ““-1 - - 2”



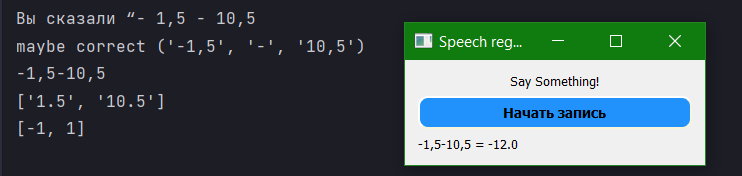


Рис.21 – Результат работы команды “- 1,5 - 10,5”

Поиск команды по вычислению двух чисел осуществляется через регулярные выражение, если в фразе есть заданный шаблон, то идет обработка метода “sub” в котором идет учет возможности наличия знака перед числами (он определяется также через регулярные выражения), затем результат записывается при помощи форматирования.

**Ссылка на GitHub:**

<https://github.com/Diacious/Voice_Assistant>

tp