Создание голосового помощника с помощью языка программирования Python.

Голосовые помощники – эффективные инструменты, которые упрощают нашу жизнь и делают ее более удобной. С развитием технологий связанных с машинным обучением, голосовые помощники стали стандартом на мобильных устройствах, планшетах, умных часах, а также в смарт домах и других IoT-устройствах. Если вы хотите создать свой голосовой помощник, используя язык программирования Python, данный проект может оказаться для вас полезным. В этом проекте мы рассмотрим как создать голосовой помощник для вашего компьютера используя язык программирования Python.

Обоснованность проблемы проекта и выделения целевой группы.

В эпоху цифровизации очень важно уметь владеть компьютером, однако для некоторых людей освоить навык владения ПК тяжело.

Целевой группой данного проекта являются пенсионеры. Данный голосовой помощник заменит ввод запросов с клавиатуры на голосовой, что является значительным упрощением задачи для людей старшего поколения.

Однако данный голосовой помощник может пригодиться людям всех возрастов и сфер деятельности, желающих оптимизировать свой рабочий процесс и облегчить поиск информации в сети.

Цели.

Изучить нужные библиотеки для создания голосового ассистента.

Разработать голосовой помощник с помощью азов программирования, который бы мог отвечать на вопросы и выполнять команды пользователей, с помощью языка программирования Python.

План реализации проекта.

1. Изучение библиотек, используемых для работы с голосом: speech recognition, pyttsx3.
2. Изучение GUI PyQT5 для создания приложения с интуитивно понятным интерфейсом.
3. Создание набросков дизайна голосового ассистента в Adobe Photoshop.
4. Создание многооконного интерфейса, его подключение к единой обработке событий.
5. Создание модуля распознавания речи и транскрибации текста с помощью библиотеки speech recognition.
6. Разработка модуля генерации речи с помощью библиотеки pyttsx3.
7. Разработка функций, обрабатывающих строку, полученную в результате использования библиотеки speech recognition на голосовой ввод пользователя
8. Тестирование готового продукта.

Качество реализации проекта:

- Быстрый и точный ответ на запросы пользователей.
- Легкость в использовании.
- Широкий функционал.
- Практическая польза от готового продукта.

Стратегия развития:

- Улучшение синтеза речи и распознавания голоса.
- Добавление новых функций, таких как распознавание настроения пользователя и ответы с учетом этого настроения.
- Расширение базы знаний и улучшение ответов на запросы.
- Интеграция с крупными мессенджерами для возможности отправки сообщений, составления заметок и т.д.
- Подключение к нейросетям, что увеличит функционал голосового помощника в несколько раз: например к нейросети от Сбера «ГигаЧат».

Приложение:

Код программы:

1) **Основное окно**

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(600, 600)
        MainWindow.setMinimumSize(QtCore.QSize(600, 600))
        MainWindow.setMaximumSize(QtCore.QSize(600, 600))
        MainWindow.setStyleSheet("background-color: qlineargradient(spread:pad, x1:1, y1:1, x2:0, y2:0, stop:0.689266 rgba(247, 85, 193, 255), stop:1 rgba(255, 255, 255, 255));\n"
"\n"
"QPushButton: {\n"
"    background-color: white;\n"
"}\n"
"\n"
"QPushButton:hover {\n"
"    background-color: #666;\n"
"}\n"
"\n"
```

```
"\n"
"QPushButton:pressed {\n"
"    background-color:  #888;\n"
"}\n"
"\n"
"font-family: Arial Black;")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.lbl_middle = QtWidgets.QLabel(self.centralwidget)
        self.lbl_middle.setGeometry(QtCore.QRect(0, -20, 931, 761))
        self.lbl_middle.setText("")
        self.lbl_middle.setPixmap(QtGui.QPixmap("../обои/новый.png"))
        self.lbl_middle.setObjectName("lbl_middle")
        self.btn_start = QtWidgets.QPushButton(self.centralwidget)
        self.btn_start.setGeometry(QtCore.QRect(210, 460, 181, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,
QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.btn_start.sizePolicy().hasHeightForWidth())
        self.btn_start.setSizePolicy(sizePolicy)
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(235, 199, 185))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
self.btn_start.setPalette(palette)
font = QtGui.QFont()
font.setFamily("Arial Black")
font.setPointSize(18)
font.setBold(True)
font.setWeight(75)
self.btn_start.setFont(font)
self.btn_start.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
```

```
        self.btn_start.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.btn_start.setStyleSheet("background-color: #ebc7b9;\n"
"border: 2px solid #ebc7b9;\n"
"border-radius: 30;\n"
"color: white\n"
"\n"
"")
        self.btn_start.setObjectName("btn_start")
        self.btn_help = QtWidgets.QPushButton(self.centralwidget)
        self.btn_help.setGeometry(QtCore.QRect(560, 10, 31, 31))
        font = QtGui.QFont()
        font.setFamily("Arial Black")
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.btn_help.setFont(font)
        self.btn_help.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.btn_help.setStyleSheet("background-color: #ebc7b9;\n"
"border: 2px solid #ebc7b9;\n"
"border-radius: 10;\n"
"color: white\n"
"\n"
"")
        self.btn_help.setObjectName("btn_help")
        self.lbl_name = QtWidgets.QLabel(self.centralwidget)
        self.lbl_name.setEnabled(True)
        self.lbl_name.setGeometry(QtCore.QRect(170, 120, 251, 31))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
```

```
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.lbl_name.sizePolicy().hasHeightForWidth())
        self.lbl_name.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Arial Black")
        font.setPointSize(17)
        font.setBold(False)
        font.setWeight(50)
        self.lbl_name.setFont(font)
        self.lbl_name.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.lbl_name.setAutoFillBackground(False)
        self.lbl_name.setStyleSheet("color: white;\n"
"border-radius: 15;\n"
"background-color:#ebc7b9")
        self.lbl_name.setObjectName("lbl_name")
        self.btn_change_window = QtWidgets.QPushButton(self.centralwidget)
        self.btn_change_window.setGeometry(QtCore.QRect(490, 570, 101, 23))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.btn_change_window.setFont(font)

self.btn_change_window.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.btn_change_window.setStyleSheet("background-color: #ebc7b9;\n"
```

```
"border: 2px solid #ebc7b9;\n"
"border-radius: 10;\n"
"color: white\n"
"")
        self.btn_change_window.setObjectName("btn_change_window")
        self.lbl_golubi1 = QtWidgets.QLabel(self.centralwidget)
        self.lbl_golubi1.setGeometry(QtCore.QRect(-410, 450, 591, 601))
        self.lbl_golubi1.setText("")
        self.lbl_golubi1.setPixmap(QtGui.QPixmap("../обои/голуби справа.png"))
        self.lbl_golubi1.setObjectName("lbl_golubi1")
        self.lbl_golubi2 = QtWidgets.QLabel(self.centralwidget)
        self.lbl_golubi2.setGeometry(QtCore.QRect(470, -520, 591, 721))
        self.lbl_golubi2.setStyleSheet("background-color: qlineargradient(spread:pad,
x1:0.358, y1:0.994318, x2:0, y2:0.233, stop:0.397727 rgba(255, 82, 198, 255), stop:1
rgba(255, 255, 255, 255));\n"
"background-color: qlineargradient(spread:pad, x1:0.392, y1:0.988636, x2:0,
y2:0.233, stop:0.397727 rgba(255, 82, 198, 255), stop:1 rgba(255, 255, 255, 255));")
        self.lbl_golubi2.setText("")
        self.lbl_golubi2.setPixmap(QtGui.QPixmap("../обои/голубь2.png"))
        self.lbl_golubi2.setObjectName("lbl_golubi2")
        self.lbl_middle.raise_()
        self.btn_start.raise_()
        self.lbl_name.raise_()
        self.btn_change_window.raise_()
        self.lbl_golubi1.raise_()
        self.lbl_golubi2.raise_()
        self.btn_help.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
```

```
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Голосовой помощник Василий"))
        self.btn_start.setText(_translate("MainWindow", "Пуск"))
        self.btn_help.setText(_translate("MainWindow", "?"))
        self.lbl_name.setText(_translate("MainWindow", "Привет, я Василий."))
        self.btn_change_window.setText(_translate("MainWindow", "Фоновый режим"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

2) Создание окна для фонового режима.

```
from PyQt5 import QtCore, QtGui, QtWidgets
```

```
class Ui_dialog(object):
    def setupUi(self, dialog):
        dialog.setObjectName("dialog")
        dialog.resize(94, 50)
        dialog.setMinimumSize(QtCore.QSize(94, 50))
        dialog.setMaximumSize(QtCore.QSize(94, 50))
        dialog.setStyleSheet("background-color: qlineargradient(spread:pad, x1:1, y1:1,
x2:0, y2:0, stop:0.689266 rgba(247, 85, 193, 255), stop:1 rgba(255, 255, 255,
255));\n"
"v\n"
"\n"
"QPushButton:hover {\n"
"    background-color: #666;\n"
"}\n"
"\n"
"\n"
"QPushButton:pressed {\n"
"    background-color:  #888;\n"
"}\n"
"\n"
"font-family: Arial Black;")
        self.btn_start_dialog = QtWidgets.QPushButton(dialog)
        self.btn_start_dialog.setGeometry(QtCore.QRect(10, 10, 71, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(11)
        font.setBold(True)
```

```
        font.setWeight(75)
        self.btn_start_dialog.setFont(font)
        self.btn_start_dialog.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.btn_start_dialog.setStyleSheet("background-color: #ebc7b9;\n"
"border: 2px solid #ebc7b9;\n"
"border-radius: 10;\n"
"color: white\n"
"")
        self.btn_start_dialog.setObjectName("btn_start_dialog")


        self.retranslateUi(dialog)
        QtCore.QMetaObject.connectSlotsByName(dialog)

    def retranslateUi(self, dialog):
        _translate = QtCore.QCoreApplication.translate
        dialog.setWindowTitle(_translate("dialog", "Голосовой помощник Василий"))
        self.btn_start_dialog.setText(_translate("dialog", "Пуск"))



if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    dialog = QtWidgets.QDialog()
    ui = Ui_dialog()
    ui.setupUi(dialog)
    dialog.show()
    sys.exit(app.exec_())
```

3) Создание help-окна, с информацией о программе.

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(400, 500)
        Form.setStyleSheet("background-color: qlineargradient(spread:pad, x1:1, y1:1,
x2:0, y2:0, stop:0.689266 rgba(247, 85, 193, 255), stop:1 rgba(255, 255, 255, 255));")
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(120, 10, 161, 41))
        font = QtGui.QFont()
        font.setFamily("Arial Black")
        font.setPointSize(13)
        font.setBold(False)
        font.setWeight(50)
        self.label.setFont(font)
        self.label.setStyleSheet("background-color: #ebc7b9;\n"
"border: 2px solid #ebc7b9;\n"
"border-radius: 20;\n"
"color: white")
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(20, 70, 361, 421))
        font = QtGui.QFont()
        font.setFamily("Arial Black")
        font.setPointSize(11)
        font.setBold(False)
```

```
        font.setUnderline(False)
        font.setWeight(50)
        font.setKerning(True)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("background-color: #ebc7b9;\n"
"border: 2px solid #ebc7b9;\n"
"border-radius: 50;\n"
"color: white")
        self.label_2.setAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
        self.label_2.setWordWrap(True)
        self.label_2.setObjectName("label_2")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Голосовой помощник Василий"))
        self.label.setText(_translate("Form", "Что это такое?"))
        self.label_2.setText(_translate("Form", "Голосовой помощник Василий -
```
виртуальный голосовой помощник. Распознаёт естественную речь, даёт ответы на вопросы пользователя. Не все пользователи компьютеров в совершенстве владеют ими, что в особенности относится к старшему поколению. Данный голосовой помощник поможет этим людям с помощью голосового ввода управлять компьютером, открывать различные сайты. Помимо этого, приложение должно помочь сократить время выполнения различных действий, так как пользователь может совмещать несколько действий одновременно.

Помощник постоянно совершенствуется, планируется расширение функционала."))

4) Основная программа.

```
import configparser
import os
from PyQt5.QtMultimedia import QSound
from PyQt5 import QtCore, QtGui, QtWidgets
from mainwindow import Ui_MainWindow
from help import Ui_Form
from smallwindow import Ui_dialog
from random import*
from datetime import*
import datetime
import speech_recognition as sr
import webbrowser
import sys
import pyttsx3
import pywhatkit
from pyowm import OWM
from transliterate import translit
import clipboard

r=sr.Recognizer()
listener=sr.Recognizer()
engine = pyttsx3.init()
voices=engine.getProperty("voices")
engine.setProperty("voice",voices[0].id)
```

```
def talk(text):
    engine.say(text)
    engine.runAndWait()

def take_command():
    global listener
    global voice
    global MainWindow



    QSound.play("C:/Users/Стас/Desktop/приложения на питоне практика/голосовой помощник/звуки/denis-dar-repin-z_uk-_sply_ayuschego-u_edomleniya-vkontakte.wav")
    try:
        with sr.Microphone() as source:
            voice = listener.listen(source)
            command = listener.recognize_google(voice, language="ru")
    except:
        talk("Извините, произошла ошибка. Проверьте подключение микрофона, приложение выключится через пару секунд")

    return command

def answer():
    global voice
    global engine
```

```
    settings = configparser.ConfigParser()
    settings.read('settings.ini')
    command=take_command()


    if command in ["Сколько сейчас времени","Скажи время","Сколько времени","Который час", "Текущее время"]:
        talk(f"Сейчас {str(datetime.datetime.now())[11:13]} часов, {str(datetime.datetime.now())[14:16]} минут")


    elif command in ["Привет", "Доброе утро", "Добрый вечер", "Здравствуйте", "Здравствуй","Здарова","Алло","Слушай", "Слушай Вася", "Здорово"]:
        talk("Здравствуйте, голосовой помощник Василий к вашим услугам")


    elif command in ["Вася", "Васёк", "Васек", "Васька", "Вась","Вася"]: talk("Я вас слушаю")


    elif command in ["Выключи компьютер", "Выруби комп", "Выключи ПК", "Выключи пк", "Выруби компьютер"]:
        talk("Есть, компьютер выключится ровно через минуту")
        os.system('shutdown /s /f /t 60')


    elif command in ["Перезагрузи компьютер", "Перезагрузи комп", "Перезагрузи ПК", "Перезагрузи пк", "Перезагрузи", "перезагрузи компьютер", "перезагрузи
```

```
комп"]:
        talk("Понял, компьютер перезагрузится ровно через минуту")
        os.system('shutdown /r /f /t 60')


    elif command in ["Переведи","переведи","Вася переведи","вася
переведи","вася переведи пожалуйста","вася быстро переведи",
                     "вася как будет","как будет","что значит слово","Вася Переведи" ]:
        talk("Если вы хотите перевести с русского на английский, скажите
английский. Если же вы хотите перевести с английского на русский, скажите
русский")
        lang=str(take_command())
        if lang=="английский" or lang=="Английский" or lang=="английский язык"
or lang=="Английский язык":
            talk('Скажите слово или фразу целиком на русском, которое нужно
перевести, а я сейчас попытаюсь это сделать')
            word = take_command()

webbrowser.open(f'https://translate.google.com/?sl=ru&tl=en&text={word}&op=tran
slate')
        else:
            talk('Скажите слово или фразу целиком на английском, которое нужно
перевести, а я сейчас попытаюсь это сделать')
            command = listener.recognize_google(voice, language="en")
            word = translit(take_command(), language_code="ru", reversed=True)

webbrowser.open(f'https://translate.google.com/?sl=en&tl=ru&text={word}&op=tran
slate')
```

```
            command = listener.recognize_google(voice, language="ru")


    elif command in ["погода", "Погода", "какая сейчас погода", "Какая сейчас погода", "Что там за окном", "Погоду скажи", "Какая погода", "какая погода", "погоду скажи", "Вася погода", "Василий погода","Вася скажи погоду","Вася какая погода","Василий какая погода","Василий скажи погоду"]:
        talk("Назовите город, в котором хотите узнать погоду")
        owm = OWM('ea547cdf5e270c71fd71db1e0d41e942')
        name_city=translit(str(take_command()), language_code='ru', reversed=True).title()
        mgr = owm.weather_manager()

        observation = mgr.weather_at_place(f"{name_city}")
        w = observation.weather
        temp=((w.temperature('celsius'))['temp'])
        temp_feels_like=((w.temperature('celsius'))['feels_like'])

        talk(f"На улице {temp} градусов по Цельсию \n"
             f"Ощущается как {temp_feels_like} градусов по Цельсию")
    elif command in ["открой карту", "карта","Открой карту","карта мира","гугл карты","гугл карта","Вася открой карту",
                     "Вася карта","вася открой карту","Вася Открой Карту","Вася Открой карту", "Открой карты"]:
        talk("Понял, открываю")
        webbrowser.open(f'https://yandex.ru/maps')

    elif command in ["открой vk", "ВКонтакте", "Открой ВКонтакте", "вконтакте",
```

```
"Открой ВК"]:
        talk("Секундочку")
        webbrowser.open(f'https://vk.com')


    elif command in ["Новости", "Открой новости", "открой новости", "новости"]:
        talk("Сейчас открою")
        webbrowser.open(f'https://dzen.ru/news')


    elif command in ["создай папку","папка", "Создай папку", "Создай Папку"]:
        talk("Скажите название папки, которую вы бы хотели создать")
        call_of_dir=take_command()
        new_adress=f"{call_of_dir}"
        if not os.path.exists(new_adress):
            os.makedirs(new_adress)
            talk(f"Папка {call_of_dir} успешно создана")
        else: talk(f"Папка с именем {call_of_dir} уже существует")


    elif command in ["открой ютуб", "открой ютюб", "ютюб", "ютуб",
"youtube","Открой YouTube", "YouTube"]:
        talk("Хорошо, открываю")
        webbrowser.open(f'https://www.youtube.com')

    elif "пароль" in command or "Пароль" in command:
        talk("Генерирую максимально защищенный пароль")
        password=""
```

```
        chars = '+-
/*!&$#?=@<>abcdefghijklnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890'
        for i in range(32):
            password+=chars[randint(0,74)]
        clipboard.copy(password)
        talk("Пароль сгенерирован и скопирован в буфер обмена")

    elif "песня" in command or "песню" in command:
        talk("Скажите название песни")
        song_name=str(take_command())
        pywhatkit.playonyt(song_name)
        talk("Включаю, приятного прослушивания")

    elif "шутка" in command or "пошути" in command or "сарказм" in "command" or
"шутку" in command or "Пошути" in command:
        talk(choice(["идёт как-то медведь по лесу", "колобок повесился", "русалка
на шпагат села", "вышел пошутить смешные",

"я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@я@ты@
я@ты@я@ты@","азат умный","зачем ты это слушаешь"]))

    elif "фильм" in command:
        talk("Скажите название фильма")
        name_film=str(take_command())

webbrowser.open(f'https://yandex.ru/search/?text=смотреть+фильм+{name_film}+о
нлайн+бесплатно')
```

```
    elif "Что такое" in command or "Расскажи про" in command or "Почему" in command or "Как" in command or "Зачем":
        talk("Ищу в интернете")

webbrowser.open(f"https://yandex.ru/search/?clid=2358536&text={command}&l10n=ru&lr=10987")



    else:
        if command!="": talk("Ваша команда не распознана, повторите ещё раз")









app = QtWidgets.QApplication(sys.argv)


MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
```

```
MainWindow.show()

QSound.play("C:/Users/Стас/Desktop/приложения на питоне практика/голосовой помощник/звуки/lantasy-harp-stinger_fjxz_seu.wav")

def OpenHelpWindow():
    global Form
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()

def OpenSmallWindow():
    global dialog
    dialog = QtWidgets.QDialog()
    ui = Ui_dialog()
    ui.setupUi(dialog)
    MainWindow.close()
    dialog.show()
    ui.btn_start_dialog.clicked.connect(answer)


ui.btn_help.clicked.connect(OpenHelpWindow)
ui.btn_change_window.clicked.connect(OpenSmallWindow)
ui.btn_start.clicked.connect(answer)


sys.exit(app.exec_())
```