Научно-исследовательская работа

Информатика

«Python.

Голосовой ассистент для ПК»

***Выполнил(а)****:*

*Бадретдинов Тимур Рустемович*

учащий(ая)ся 11А класса

МАОУ «Гимназия №115» ГО г.Уфа Республики Башкортостан

***Руководитель****:*

*Петов Олег Владимирович*

Преподаватель информатики

МАОУ «Гимназия №115» ГО г.Уфа Республики Башкортостан

Оглавление

Введение

**Глава 1. I Этап. Изучение материалов по обозначенной теме**

**Глава 2. II Этап. Отбор материалов для эксперимента**

**Глава 3. III Этап. Проведение эксперимента**

**Глава 4. IV Этап. Обработка результатов**

**Глава 5. Приложение**

**Заключение**

**ВЕДЕНИЕ.**

**Актуальность**. Изучение структуры и принципов разработки компьютерных приложений не только позволяет расширить знания об интегрированных средах программирования, но способствует активизации образовательной деятельности школьников в различных предметных областях. Полученные навыки и умения в дальнейшем можно применить при разработке обучающих или образовательных приложений.

Изучив возможности алгоритмических языков в области, можно получить неоценимые практические навыки в программировании.

Все вышеперечисленные навыки будут неоценимы при учебе в школе, университете и при поступлении на работу. А созданное приложение будет верным помощником каждому пользователю ПК.

**Цель исследования**. Изучение и исследование инструментов в области разработки приложений для создания приложений .

**Объект исследования** – современные программные инструменты, структура, возможности и разработка обучающих приложений для ПК.

**Предмет исследования** – возможности языка Python в области разработки голосового ассистента.

**Задачи исследования:**

* познакомиться с возможностями использования языка Python, как инструмента разработки интерактивных приложений;
* познакомиться со структурой обучающих приложений;
* провести эксперимент по разработке приложения на языке Python;
* проанализировать способы реализации универсальности программы.

**Методы:**

* Сравнение.
* Эксперимент.
* Анализ.
* Моделирование.
* Обобщение.

**ГЛАВА 1. I Этап. Изучение материалов по обозначенной теме**

**Голосовой помощник** — программное обеспечение, позволяющее управлять мобильным устройством или компьютером посредством голосовых команд. Современный голосовой помощник (или "виртуальный ассистент" от англ. "virtual assistant") оказывает реальную помощь в различных областях информационных технологий- поиск информации в Интернете, запуск системных функции и приложения, и при этом выступать в роли виртуального собеседника.

Помимо распознавания голоса, ассистент способен также озвучивать текстовую информацию (например, результаты поиска), "говорить" с человеком или "общаться" с ним в виде текстового чата. Полноценный диалог ни с одним из голосовых помощников, к сожалению, пока невозможен.

На сегодняшний день наблюдается тенденция к закреплению за популярными операционными системами собственных голосовых помощников. Так, на iOS штатным ассистентом является программа Siri, на Android — Google Assistant, на Windows — Cortana. Об этих и некоторых других современных "умных" помощниках и пойдёт речь в статье ниже.

Голосовые ассистенты встроены в компьютеры, планшеты, телефоны, умные часы, умные колонки и даже в автомобили. Диалог с голосовым помощником осуществляется исключительно голосом, без использования рук, не нажимая ни на какие кнопки. Это принципиально новый способ взаимодействия человека и устройства, он довольно похож на общение между людьми.

Но все ли так хорошо как кажется на первый взгляд? Утечка личных и корпоративных тайн в руки недоброжелателей — это раз. Диктуя вслух цифры полученного в SMS кода авторизации или реквизиты банковских карт, когда заполняете онлайн-формы, вы тем самым даете возможность мошенникам.

Шансов, что ваш разговор не будет услышан и оцифрован, современные технологии не оставляют. Умные колонки «слышат» издалека в шумной обстановке, даже при играющей музыке. Да и говорить как-то особенно четко не обязательно обыкновенный помощник Google в планшете иногда лучше родителей понимает произношение трехлетнего ребенка.

Голосовой ассистент должен обладать уникальными чертами отличающимся функционалом. Это следует из того, что разработчики используют свои подходы к разработке и разные алгоритмы.

К основным технологиям можно отнести следующие:

* активация по голосу (Voice Activation),
* автоматическое распознавание речи (Automatic Speech Recognition),
* синтез речи (Text-To-Speech),
* голосовая биометрия (Voice Biometrics), т.е. распознавание пола или возраста говорящего, например женщины, мужчины, детей и т.д., а также диалоговый менеджер (Dialog Manager),
* понимание естественного языка (Natural Language Understanding),
* распознавание именованных сущностей (Named Entity Recognition) [8-10, 12].

Для разработки голосового ассистента необходим соответствующий софт. В своей работе я буду использовать Python.

**Pyhton.** Pyhton — среда разработки, использует язык программирования Pyhton (начиная с 7 версии язык в среде именуется Pyhton[2], ранее — Object Pascal), разработанный фирмой Borland и изначально реализованный в её пакете Borland Pyhton, от которого и получил в 2003 году своё нынешнее название. Object Pascal по сути является наследником языка Pascal с объектно-ориентированными расширениями.

Pyhton — это среда быстрой разработки, в которой в качестве языка программирования используется язык Pyhton. Язык Pyhton — строго типизированный объектно-ориентированный язык, в основе которого лежит хорошо знакомый программистам Object Pascal.

Pyhton — это комбинация нескольких важнейших технологий:

высокопроизводительный компилятор в машинный код;

– объектно-ориентированная модель компонент;

– визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;

– масштабируемые средства для построения баз данных.

Borland Pyhton 8 Studio позволяет создавать самые различные программы: от простейших однооконных приложений до программ управления распределенными базами. В состав пакета включены разнообразные утилиты, обеспечивающие работу с базами данных, XML-документами, создание справочной системы, решение других задач. Отличительной особенностью седьмой версии является поддержка технологии .NET.

Основной упор модели в Pyhton делается на то ,чтобы максимально производительно использовать код.. А так же возможность создавать свои собственные объекты.

В стандартную поставку Pyhton входят основные объекты из 270 базовых классов. На этом языке очень удобно писать, как приложения к базам данных, так даже и игровые программы. Если принять во внимание и удобный интерфейс для создания графических оболочек, то можно с уверенностью заявить что язык Pyhton – это очень доступный для понимания, но в то же время и очень мощный язык программирования.

Первая версия полноценной среды разработки Pyhton для .NET — Pyhton 8. Она позволяла писать приложения только для .NET. Среда также позволяет создавать .NET-приложения на C# и Win32-приложения на C++. Pyhton 2006 содержит функции для написания обычных приложений с использованием библиотек VCL и CLX. Pyhton 2006 поддерживает технологию MDA с помощью ECO (Enterprise Core Objects) версии 3.0.

**Глава 2. Отбор материалов для эксперимента**

В настоящее время можно выделить несколько типов языков программирования. Признаков их классификации служит принадлежность их к одному из стилей: процедурный, функциональный, логический, объектно-ориентированный.

Основная цель ООП – повышение эффективности разработки программ. Идеи ООП оказались плодотворными и нашли применение не только в языках программирования, но и в других областях Computer Science, например, в области разработки операционных систем.

Концепция объектно-ориентированного программирования подразумевает, что основой управления процессом реализации программы является передача сообщений объектам. Поэтому объекты должны определяться совместно с сообщениями, на которые они должны реагировать при выполнении программы. В этом состоит главное отличие ООП от процедурного программирования.

Pyhton – это потомок среды программирования Turbo Pascal. Система визуального объектно-ориентированного проектирования Pyhton позволяет:

Создавать законченные приложения для Windows самой различной направленности.

Быстро создавать профессионально выглядящий оконный интерфейс для любых приложений; интерфейс удовлетворяет всем требованиям Windows и автоматически настраивается на ту систему, которая установлена, поскольку использует функции, процедуры и библиотеки Windows.

Создавать свои динамически присоединяемые библиотеки компонентов, форм, функций, которые потом можно использовать из других языков программирования.

Создавать мощные системы работы с базами данных любых типов.

Формировать и печатать сложные отчеты, включающие таблицы, графики и т.п.

Создавать справочные системы, как для своих приложений, так и для любых других.

Создавать профессиональные программы установки для приложений Windows, учитывающие всю специфику и все требования операционной системы.

Интегрированная среда разработки Pyhton – это среда, в которой есть все необходимое для проектирования, запуска и тестирования создаваемых приложений. Большинство версий Pyhton выпускается в нескольких вариантах: а) стандартная, б) профессиональная версия, в) разработка баз данных предметных областей. Эти варианты различаются, в основном разным уровнем доступа к системам управления базами данных. Последние два варианта являются наиболее мощными в этом отношении. Библиотеки компонентов в различных вариантах практически одинаковы.

Ниже полосы главного меню расположены две **инструментальные панели**. Левая панель (состоящая, в свою очередь, из трех панелей) содержит два ряда кнопок, дублирующих некоторые наиболее часто используемые команды меню (открыть, сохранить, сохранить все и т.д.). Правая панель содержит панель библиотеки визуальных компонентов (или палитра). **Палитра** компонентов содержит ряд страниц, закладки которых видны в ее верхней части. Страницы сгруппированы в соответствии с их смыслом и назначением.

Под палитрой компонентов располагается **окно формы** с размещенными на ней компонентами. Форма является основой почти всех приложений Pyhton. Форму можно понимать как типичное окно Windows. Она обладает теми же свойствами, что и другие окна. В основном поле окна слева находится **окно Инспектора объектов**, с помощью которого в дальнейшем можно задавать свойства компонентов и обработчики событий. Каждый компонент имеет свой набор свойств, который соответствует назначению этого компонента.

Одним из наиболее важных элементов среды Pyhton является окно **Редактора кода**. Оно располагается ниже окна формы, обычно при первом взгляде на экран невидимо, т. к. его размер равен размеру формы и окно Выше окна Инспектора объектов расположено окно **Дерево объектов**, которое отображает структуру компонентов приложения с точки зрения их принадлежности друг к другу.

Программа, создаваемая в среде Pyhton в процессе проектирования приложения, основана на модульном принципе. Главная программа состоит из объявления списка используемых модулей и нескольких операторов, создающих объекты для необходимых форм и запускающих приложение на выполнение. Модульность очень важна для создания надежных и относительно легко модифицируемых и сопровождаемых приложений. Четкое соблюдение принципов модульности в сочетании с принципом скрытия информации позволяет производить модификации внутри любого модуля, не затрагивая при этом остальных модулей и главную программу.

В процессе проектирования Pyhton автоматически создает код головной программы и отдельных модулей. В модули вводятся собственные коды, создавая обработчики различных событий. Но головную программу, как правило, не приходится модифицировать и даже просматривать ее текст (только в исключительных случаях).

**Глава 3. Проведение эксперимента**

В данной главе кратко опишем алгоритм разработки голосового ассистента для нашего компьютера.

Шаг №1. Импорт всех необходимых для полноценной работы библиотек

import speech\_recognition as sr

import os

import sys

import re

import webbrowser

import smtplib

import requests

import subprocess

import youtube\_dl

import vlc

import urllib

import urllib2

import json

from bs4 import BeautifulSoup as soup

from urllib2 import urlopen

import wikipedia  
Этот перечень можно дополнить в зависимости от решения конкретной задачи.

Шаг №2. Написание программного кода для логики каждой из вышеперечисленных функций.

Шаг №3. Разработать метод интерпретации голосового ответа и метода перевода текста в звук.  
def myCommand():

r = sr.Recognizer()

with sr.Microphone() as source:

print('Say something...')

r.pause\_threshold = 1

r.adjust\_for\_ambient\_noise(source, duration=1)

audio = r.listen(source)

try:

command = r.recognize\_google(audio).lower()

print('You said: ' + command + '\n')

#loop back to continue to listen for commands if unrecognizable speech is received

except sr.UnknownValueError:

print('....')

command = myCommand();

return command

def sofiaResponse(audio):

print(audio)

for line in audio.splitlines():

os.system("say " + audio)

Шаг №4. Написание нескольких условных алгоритмов для каждой из функции. А конкретно:

Открыть любой web-сайт в браузере

Работа с электронной почтой

Новостная лента

Запуск приложений

Воспроизведение аудио-видеофайлов

Пуск и завершение работы.

Шаг №5. Исправление ошибок, настройка приложения, адаптация под различные операционные системы.

**4.3.Заключение**

На данном этапе разработки приложения были значительно доработаны некоторые функции по сравнению с первоначальной версией. Отредактирован код для текста, его формы и цвета. Проведено тестирование приложения, оптимизирована его цветовая гамма. Протестировав голосовой ассистент на нескольких операционных системах линейки Windows сделан вывод о его корректной работе.

Приложение работает без сбоев. Тестирование корректно компьютерах с различной аппаратной частью показало полную работоспособность приложения.

В данный ведется доработка приложения в области его пользовательского интерфейса. Проводятся исследования в области интегрирования в ассистент возможности пользовательских настроек.

На основании выводов по завершению разработки проекта, было установлено, что интерактивное приложение является довольно сложной системой, разработка которой занимает много времени, а отладка и компоновка интерфейса должна проводиться с непосредственным участием тестирующих людей.

Работа была проведена на хорошем уровне и достигнута цель исследования. Полученные результаты позволяют усовершенствовать данное приложение в дальнейшем.

**Глава 5. Приложение**

import os, time, datetime, logging, webbrowser, subprocess, re, sys, html2text, pygame, urllib.request

from urllib import request

from urllib.parse import quote

from mywindow import \*

from PyQt5 import QtCore, QtGui, QtWidgets

import speech\_recognition as sr

from gtts import gTTS

from pygame import mixer

mixer.init()

def setMoveWindow(widget):

"""Позволяет перемещать окно ухватившись не только за заголовок, а за произвольный виджит (widget) """

win = widget.window()

cursor\_shape = widget.cursor().shape()

move\_source = getattr(widget, "mouseMoveEvent")

press\_source = getattr(widget, "mousePressEvent")

release\_source = getattr(widget, "mouseReleaseEvent")

def move(event):

if move.b\_move:

x = event.globalX() + move.x\_korr - move.lastPoint.x()

y = event.globalY() + move.y\_korr - move.lastPoint.y()

win.move(x, y)

widget.setCursor(QtCore.Qt.SizeAllCursor)

return move\_source(event)

def press(event):

if event.button() == QtCore.Qt.LeftButton:

# Корекция геометрии окна: учитываем размеры рамки и заголовока

x\_korr = win.frameGeometry().x() - win.geometry().x()

y\_korr = win.frameGeometry().y() - win.geometry().y()

# Корекция геометрии виджита: учитываем смещение относительно окна

parent = widget

while not parent == win:

x\_korr -= parent.x()

y\_korr -= parent.y()

parent = parent.parent()

move.\_\_dict\_\_.update({"lastPoint": event.pos(), "b\_move": True, "x\_korr": x\_korr, "y\_korr": y\_korr})

else:

move.\_\_dict\_\_.update({"b\_move": False})

widget.setCursor(cursor\_shape)

return press\_source(event)

def release(event):

move.\_\_dict\_\_.update({"b\_move": False})

widget.setCursor(cursor\_shape)

return release\_source(event)

setattr(widget, "mouseMoveEvent", move)

setattr(widget, "mousePressEvent", press)

setattr(widget, "mouseReleaseEvent", release)

move.\_\_dict\_\_.update({"b\_move": False})

return widget

class MyWin(QtWidgets.QMainWindow):

def \_\_init\_\_(self, parent=None):

global sr

QtWidgets.QWidget.\_\_init\_\_(self, parent)

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

qss\_file = open('style\_file.qss').read()

self.setStyleSheet(qss\_file)

self.ui.pushButton.clicked.connect(self.mycommand)

self.ui.toolButton\_2.clicked.connect(self.myexit)

self.ui.toolButton.clicked.connect(self.mymin)

self.\_recognizer = sr.Recognizer()

self.\_microphone = sr.Microphone()

now\_time = datetime.datetime.now()

self.\_mp3\_name = now\_time.strftime("%d%m%Y%I%M%S") + ".mp3"

self.\_mp3\_nameold = '111'

# Функция для запуска команд с командной строки Windows

def osrun(self, cmd):

PIPE = subprocess.PIPE

p = subprocess.Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=subprocess.STDOUT)

# Функция открывает в браузере определенный URL и произносит фразу

def openurl(self, url, ans):

webbrowser.open(url)

self.say(str(ans))

while pygame.mixer.music.get\_busy():

time.sleep(0.1)

# Функция произносит вслух фразу

def say(self, phrase):

tts = gTTS(text=phrase, lang="ru")

tts.save(self.\_mp3\_name)

mixer.music.load(self.\_mp3\_name)

mixer.music.play()

if (os.path.exists(self.\_mp3\_nameold)):

os.remove(self.\_mp3\_nameold)

now\_time = datetime.datetime.now()

self.\_mp3\_nameold = self.\_mp3\_name

self.\_mp3\_name = now\_time.strftime("%d%m%Y%I%M%S") + ".mp3"

# Функция чистит фразу от ключевых слов

def cleanphrase(self, statement, spisok):

for x in spisok:

statement = statement.replace(x, '')

statement = statement.strip()

return statement

# Удаляем лишний mp3 файл

def \_clean\_up(self):

def clean\_up():

os.remove(self.\_mp3\_name)

# Функция выдает список url из выдачи поисковика по запросу z

def mysearch(self, z):

s = 'http://go.mail.ru/search?fm=1&q=' + quote(z)

doc = urllib.request.urlopen(s).read().decode('cp1251', errors='ignore')

o = re.compile('"url":"(.\*?)"')

l = o.findall(doc)

sp = []

for x in l:

if ((x.rfind('youtube') == -1) and (x.rfind('yandex') == -1) and (x.rfind('mail.ru') == -1) and (

x.rfind('.jpg') == -1) and (x.rfind('.png') == -1) and (x.rfind('.gif') == -1)):

sp.append(x)

sp = dict(zip(sp, sp)).values()

sp1 = []

for x in sp: sp1.append(x)

return sp1

# Функция выдает список текстов со страниц из списка переданнных в неё url

def gettexts(self, urls):

urls2 = []

urls2.append(urls[0])

urls2.append(urls[1])

texts = []

for s in urls2:

doc = urllib.request.urlopen(s).read().decode('utf-8', errors='ignore')

h = html2text.HTML2Text()

h.ignore\_links = True

h.body\_width = False

h.ignore\_images = True

doc = h.handle(doc)

summa = ""

ss = doc.split("\n")

for xx in ss:

xx = xx.strip()

if ((len(xx) > 50) and (xx.startswith('&') == False) and (xx.startswith('>') == False) and (

xx.startswith('\*') == False) and (xx.startswith('\\') == False) and (

xx.startswith('<') == False) and (xx.startswith('(') == False) and (

xx.startswith('#') == False) and (

xx.endswith('.') or xx.endswith('?') or xx.endswith('!') or xx.endswith(';'))):

summa = summa + xx + "\n \n"

if (len(summa) > 500):

texts.append(summa)

return texts

def myexit(self):

mixer.stop()

mixer.quit()

files = os.listdir()

print(files)

files = filter(lambda x: x.endswith('.mp3'), files)

for f in files:

if (os.path.exists(f)): os.remove(f)

if (os.path.exists('index.html')): os.remove('index.html')

sys.exit()

def mymin(self):

self.showMinimized()

def mycommand(self):

print("Скажи что - нибудь!")

with self.\_microphone as source:

audio = self.\_recognizer.listen(source)

print("Понял, идет распознавание...")

try:

statement = self.\_recognizer.recognize\_google(audio, language="ru\_RU")

statement = statement.lower()

print("Вы сказали: {}".format(statement))

# Здесь идут команды для распознавания

mflag = 0

if ((statement.find("калькулятор") != -1) or (statement.find("calculator") != -1)):

self.osrun('calc')

mflag = 1

if ((statement.find("блокнот") != -1) or (statement.find("notepad") != -1)):

self.osrun('notepad')

mflag = 1

if ((statement.find("paint") != -1) or (statement.find("паинт") != -1)):

self.osrun('mspaint')

mflag = 1

if ((statement.find("browser") != -1) or (statement.find("браузер") != -1)):

self.openurl('http://google.ru', 'Открываю браузер')

mflag = 1

# Команды для открытия URL в браузере

if (((statement.find("youtube") != -1) or (statement.find("youtub") != -1) or (

statement.find("ютуб") != -1) or

(statement.find("you tube") != -1)) and (statement.find("смотреть") == -1)):

self.openurl('http://youtube.com', 'Открываю ютуб')

mflag = 1

if (((statement.find("новости") != -1) or (statement.find("новость") != -1) or (

statement.find("новасти") != -1))

and ((statement.find("youtube") == -1) and (statement.find("youtub") == -1) and (

statement.find("ютуб") == -1)

and (statement.find("you tube") == -1))):

self.openurl('https://www.youtube.com/user/rtrussian/videos', 'Открываю новости')

mflag = 1

if ((statement.find("mail") != -1) or (statement.find("майл") != -1)):

self.openurl('https://e.mail.ru/messages/inbox/', 'Открываю почту')

mflag = 1

if ((statement.find("вконтакте") != -1) or (statement.find("в контакте") != -1)):

self.openurl('http://vk.com', 'Открываю Вконтакте')

mflag = 1

# Команды для поиска в сети интернет

if ((statement.find("читать") != -1) or (statement.find("четать") != -1) or (statement.find("читати") != -1)

or (statement.find("читает") != -1) or (statement.find("чи тать") != -1) or (

statement.find("прочитать") != -1)

or (statement.find("читай") != -1) or (statement.find("прочитай") != -1) or (

statement.find("считай") != -1)):

statement = self.cleanphrase(statement,

['прочитать', 'чи тать', 'читать', 'четать', 'читати', 'читает', 'считай',

'читай', 'про'])

try:

spisok = self.mysearch('статья+' + statement)

mytext = self.gettexts(spisok)

f = open('index.html', 'w')

if (mytext[0] != ''):

sss = '<pre style="font-size: 120%; font-weight: bold; padding: 50px; background: #efefef; white-space: pre-wrap; white-space: -moz-pre-wrap; white-space: -pre-wrap; white-space: -o-pre-wrap; word-wrap: break-word;">' + str(

mytext[0]) + '</pre>'

f.write(sss)

if ((len(mytext) > 1) and (mytext[1] != '')):

sss = '<p><pre style="font-size: 120%; font-weight: bold; padding: 50px; background: #efefef; white-space: pre-wrap; white-space: -moz-pre-wrap; white-space: -pre-wrap; white-space: -o-pre-wrap; word-wrap: break-word;">' + str(

mytext[1]) + '</pre>'

f.write(sss)

f.close()

webbrowser.open('index.html')

except:

self.openurl(self.mysearch('статья+' + statement)[1], "Вот что я нашла")

mflag = 1

if ((statement.find("покажи") != -1) or (statement.find("показать") != -1)):

statement = self.cleanphrase(statement, ['покажи', 'показать'])

self.openurl(self.mysearch('статья+' + statement)[1], "Вот что я нашла")

mflag = 1

if ((statement.find("найти") != -1) or (statement.find("поиск") != -1) or (

statement.find("найди") != -1) or (statement.find("дайте") != -1) or (

statement.find("mighty") != -1)):

statement = self.cleanphrase(statement, ['найди', 'найти'])

self.openurl('https://yandex.ru/yandsearch?text=' + statement, "Я нашла следующие результаты")

mflag = 1

if (((statement.find("смотреть") != -1) or (statement.find("сматреть") != -1)) and (

(statement.find("фильм") != -1) or (statement.find("film") != -1))):

statement = self.cleanphrase(statement, ['посмотреть', 'смотреть', 'сматреть', 'хочу', 'фильм', 'film'])

self.openurl(self.mysearch('смотреть+онлайн+фильм+' + statement)[1],

"Если это нужный фильм нажмите Play")

mflag = 1

if (((statement.find("youtube") != -1) or (statement.find("ютуб") != -1) or (

statement.find("you tube") != -1)) and (statement.find("смотреть") != -1)):

statement = self.cleanphrase(statement,

['хочу', 'на ютубе', 'на ютуб', 'на youtube', 'на you tube', 'на youtub',

'youtube', 'ютуб', 'ютубе', 'посмотреть', 'смотреть'])

self.openurl('http://www.youtube.com/results?search\_query=' + statement, 'Ищу в ютуб')

mflag = 1

if ((statement.find("слушать") != -1) and (statement.find("песн") != -1)):

statement = self.cleanphrase(statement,

['песню', 'песни', 'песня', 'хочу', 'песней', 'послушать', 'слушать'])

self.openurl('https://music.yandex.ru/search?text=' + statement, "Нажмите плэй")

mflag = 1

if ((statement.find("кто ты") != -1) or (statement.find("о себе") != -1) or (statement.find("расскажи о себе") != -1)

or (statement.find("привет") != -1) or (statement.find("здравствуй") != -1)):

answer = "Я Эмили - твой голосовой помощник! Попроси меня о чём либо и я постараюсь ответить тебе"

self.say(answer)

mflag = 1

if ((statement.find("как дела") != -1)):

answer = "Хорошо. Как дела у тебя?"

self.say(answer)

mflag = 1

if ((statement.find("до свидания") != -1) or (statement.find("досвидания") != -1) or (statement.find("пока") != -1)):

answer = "Пока!"

self.say(answer)

while pygame.mixer.music.get\_busy():

time.sleep(0.1)

sys.exit()

if (mflag == 0): self.say(str('Переформулируйте команду'))

except sr.UnknownValueError:

print("Упс! Кажется, я тебя не поняла, повтори еще раз")

mflag = 0

except sr.RequestError as e:

print("Не могу получить данные от сервиса Google Speech Recognition; {0}".format(e))

if \_\_name\_\_ == "\_\_main\_\_":

app = QtWidgets.QApplication(sys.argv)

mainWin = MyWin()

setMoveWindow(mainWin)

mainWin.show()

sys.exit(app.exec\_())

**Литература**

1. [https://xreferat.com/33/1486-2-osnovy-programmirovaniya-v-srede-Pyhton-7-0.html](https://xreferat.com/33/1486-2-osnovy-programmirovaniya-v-srede-delphi-7-0.html)
2. <https://cyberleninka.ru/article/n/sravnitelnyy-analiz-programmnyh-sredstv-razrabotki-prilozheniy-i-baz-dannyh-i-individualizatsiya-uchebnogo-protsessa-ih-izucheniya>.
3. <https://geekbrains.ru>
4. <https://habr.com/ru/post/29778/>
5. <http://server.aesc.msu.ru/materials/PYTHON/pythonworldru.pdf>
6. [**https://itnan.ru/post.php?c=1&p=450224**](https://itnan.ru/post.php?c=1&p=450224)