

Содержание

| | |
|---|----|
| 1. Постановка задачи..... | 2 |
| 2. Разработка алгоритма решения задачи..... | 3 |
| 2.1 Описание Python..... | 3 |
| 2.2 Tkinter..... | 4 |
| 2.3 Создание сокетов, номер порта..... | 6 |
| 2.3.1 Создание окон, лог чата, полей..... | 6 |
| 2.3.2 Реализация передачи пользователем своего сообщения в программу по нажатию на Enter..... | 7 |
| 3. Листинг программы решения задания..... | 8 |
| 4. Описание работы..... | 14 |
| 5. Результаты выполнения работы..... | 15 |
| 6 Список используемой литературы..... | 16 |

1. Постановка задачи

Практически любой начинающий программист на Python патологически старается написать свой чат. Для начала введем в нашу задачу несколько условностей — пишем мы чат для локальной сети с разрешенными широковещательными UDP-пакетами. Для простоты решения задачи так же решим, что в качестве GUI будем использовать библиотеку Tkinter (*обычно* в дистрибутивах Linux она идет из коробки, так же и в официальной сборке Python под Windows она является стандартной). Разработать программу чата в среде Python с использованием библиотеки Tkinter.

2. Разработка алгоритма решения задачи.

2.1 Описание Python.

Python (англ.*Python* – питон, произносится ['paɪθ(ə)n] – *пайтон*; в русском языке распространено название *питон* [1]) – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Питоне организовывается в функции и классы, которые могут объединяться в модули (которые в свою очередь могут быть объединены в пакеты).

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ [2]. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные [3]. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Питона на самом Питоне, что уменьшает затраты на изменения языка и постановку экспериментов над новыми возможностями.

Python – активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с

половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет Cpython.

2.2 Tkinter.

Tkinter (от англ. Tk interface) — кросс-платформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована в том числе и на Microsoft Windows, Apple Mac OS), написанная Стином Лумхольтом и Гвидо ван Россумом. Входит в стандартную библиотеку Python.

Библиотека предназначена для организации диалогов в программе с помощью оконного графического интерфейса (GUI). В составе библиотеки присутствуют общие графические компоненты:

- Toplevel

Окно верхнего уровня (корневой виджет)

- Tk
- Frame

Рамка. Содержит в себе другие визуальные компоненты

- Label

Этикетка. Показывает некоторый текст или графическое изображение

- Entry

Поле ввода текста

- Canvas

Рисунок. Может использоваться для вывода графических примитивов, например, для построения графиков

- Button

Кнопка. Простая кнопка для выполнения команды и других действий

- Radiobutton

Переключатель. Представляет одно из альтернативных значений некоторой переменной. Обычно действует в группе. Когда пользователь выбирает какую-либо опцию, с ранее выбранного в этой же группе элемента выбор снимается.

- Chekbutton

Флажок. Кнопка, имеющая два состояния, при нажатии изменяет состояние на противоположное

- Scale

Шкала. Позволяет задать числовое значение путем перемещения движка

- Listbox

Список. Показывает список, из которых пользователь может выделить один или несколько элементов

- Scrollbar

Полоса прокрутки. Может использоваться вместе с некоторыми другими компонентами для их прокрутки

- OptionMenu
- Spinbox
- LabelFrame
- PanedWindow
- Menu

Меню. Служит для организации всплывающих и ниспадающих (pulldown) меню

- Menubutton

Кнопка-меню. Используется для организации pulldown-меню

- Message

Сообщение. Аналогично Label, но позволяет заворачивать длинные строки и легко меняет свой размер

- Text

Форматированный текст. Позволяет показывать, редактировать и форматировать текст с использованием различных стилей, а также внедрять в текст рисунки и окна.

2.3 Создание сокетов, номер порта.

Работа с сокетами в питоне не зависит от платформы (по большому счету даже под PyS60 мы получим рабочий сетевой код, по этому примеру). Для нашего чата мы решили использовать широковещательные UDP-пакеты. Одной из причин их использования была возможность отказаться от использования сервера. Необходимо принять еще одну условность в нашей программе — номер порта, и пусть он будет равен 11719, во-первых, это простое число (а ведь это уже многое). А, во-вторых, этот порт, по официальной информации IANA, не занят. У сокета мы выставили свойства `SO_REUSEADDR`(позволяет нескольким приложениям «слушать» сокет) и `SO_BROADCAST`(указывает на то, что пакеты будут широковещательные) в значение истины. На самом деле на некоторых системах возможна работа скрипта и без этих строчек, но все же лучше явно указать эти свойства. На про слушку мы подключаемся к адресу '0.0.0.0' — это означает, что прослушиваются вообще все интерфейсы. Можно указывать в поле адреса и просто пустые кавычки, но все же лучше явно указать адрес. Так же создадим широковещательную «ошибки» сети(для проверки первой программы). В отношении же передающей части, необходима лишь только опция `SO_BROADCAST`(зато теперь обязательна на всех платформах) и с помощью метода `sendto()` мы рассылаем по всей сети наши пакеты. Остановив с помощью CTRL+C сетевой флуд, перейдем к описанию интерфейса.

2.3.1 Создание окон, лог чата, полей.

Tkinter — наверное, одна из самых простых библиотек для организации

оконного интерфейса на питоне (а еще по заявлению создателя питона она является одной из самых надежных и стабильных). И для того, чтобы получить просто окно нужного нам размера и с указанным заголовком от нас требуется лишь несколько строчек кода. Значит нам нужны на окне еще и элементы — лог чата, поле, где мы укажем свой ник и текст сообщения, который мы вводим. Без кнопки отправить можно обойтись, если наше поле для сообщения будет само реагировать на нажатие клавиши Enter.

2.3.2 Реализация передачи пользователем своего сообщения в программу по нажатию на Enter.

Решение проблемы проще, чем кажется. Для лога чата можно использовать виджет Text, а для двух остальных Entry. Для того, чтобы разместить элементы на форме мы используем автоматический компоновщик pack, который будет известным только ему способом расставлять элементы, придерживаясь только явно указанных указаний. Однако, ему можно указать сторону к которой крепить виджеты, расширять ли их и в какую сторону, и еще некоторые параметры компоновки. Entry можно связать с StingVar'ами(указав в конструкторе виджетов свойство textvariable). а для запуска фоновой процедуры есть у Tkinter'a метод after(<время в мс>,<функция>). Если в конце исполнения этой процедуры указать переинициализацию ее, то процедура будет выполняться постоянно, пока запущена программа. Консоли забегало приветствие. Меняя ник мы можем убедиться, что связь между полем Entry и нашей переменной работает идеально. Самое время реализовать возможность передачи пользователем своего сообщения в программу по нажатию на Enter. Реализуется это еще проще, с помощью метода bind(<действие>, <функция>) у виджетов. Единственное, что нам нужно учесть, что функция должна принимать параметр event. За одно перенесем с консоли действие в поле лога.

3. Листинг программы решения задания.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import socket
from Tkinter import *

#Решаем вопрос с кириллицей
# Переопределении системных настроек
reload(sys)
sys.setdefaultencoding('utf-8')
#-----
#Создание экземпляров класса
tk=Tk()
#Создание сокета (Принимающий)
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST,
1)
s.bind(('0.0.0.0',11719))

#Создание сокета (отправляющий)
sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET,
socket.SO_BROADCAST,1)
# Создание переменных
text=StringVar()
name=StringVar()
name.set('HabrUser')
text.set('')
#Окно чата
tk.title('MegaChat')
tk.geometry('400x300')
log = Text(tk)
#Создание текстового поля
nick = Entry(tk, textvariable=name)
msg = Entry(tk, textvariable=text)
#Упаковка сообщения
msg.pack(side='bottom', fill='x', expand='true')
nick.pack(side='bottom', fill='x', expand='true')
```



```

log.pack(side='top', fill='both',expand='true')

#Функция постоянно запущенная
def loopproc():
    log.see(END)
#Устанавливаем неблокирующий сокет
s.setblocking(False)
# Обработка исключений
try:
#Устанавливаем размер сообщения максимально 128 Байт
    message = s.recv(128)
    log.insert(END,message+'\n')
except:
    tk.after(1,loopproc)
    return
tk.after(1,loopproc)
return

#Функция отправляющая сообщения
def sendproc(event):
    sock.sendto      (name.get()+':'+text.get(),
('255.255.255.255',11719))
    #Очищаем поле «text»
    text.set('')
    # Устанавливаем кнопку Enter, как кнопку отправляющие
сообщения
msg.bind('<Return>',sendproc)
#Оставляем фокус в поле
msg.focus_set()
tk.after(1,loopproc)
#Привязка к чату
tk.mainloop()

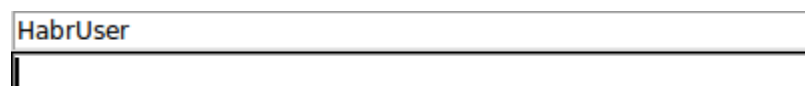
```

4. Описание работы.

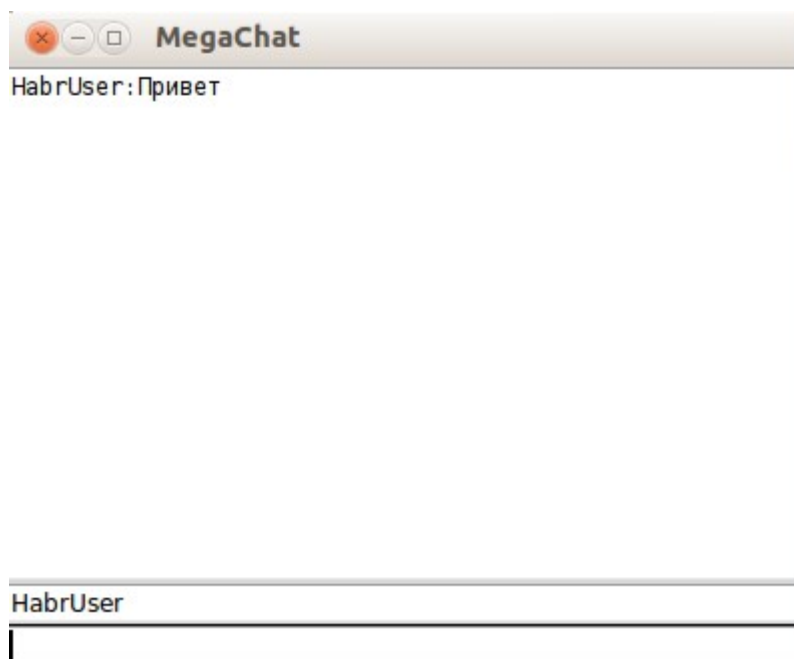
1. На первом этапе создается окошко с надписью «Ваши данные», где вам нужно ввести ваше имя вместо имени по умолчанию HabrUser.

A screenshot of a single-line text input field. The text 'HabrUser' is displayed in a light blue font, indicating it is a placeholder or default value.

2. На втором этапе создается окошко «Диалоговое окно», где вы можете в данном окне для вас отправлять сообщение собеседнику который будет отвечать вам во втором окне.

A screenshot of a dialog window. It features a text input field at the top containing the text 'HabrUser'. Below the input field is a horizontal line, and below that is a larger empty rectangular area, likely for a message or response.

3. На третьем этапе нажимаем клавишу Enter и отправляем сообщение в диалоговое окно, для его закрытия нажимаем на крестик.



5. Результаты выполнения работы.

В результате выполнения курсовой работы была осуществлена разработка чата в среде интерпретации Python с использованием библиотеки Tkinter. Проверка чата показала адекватность его работы.

6 Список используемой литературы.

1. <http://inform-uchebnik.com>
2. <http://leonov-kuchnirenko.ru>
3. <http://www.python.org/2.5/license.html>
4. <http://inform-programm.com>