Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Домашнее задание

Выполнил:
студент группы ИУ5-34Б
Ромашко Дарья
Подпись и дата:

Проверил:
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2021 г.

Задание:
Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.
Текст программы:
Файл «dbworker.py»:

```
from vedis import Vedis
import config

# Чтение значения
def get(key):
    with Vedis(config.db_file) as db:
        try:
            return db[key].decode()
        except KeyError:
            return ''


# Запись значения
def set(key, value):
    with Vedis(config.db_file) as db:
        try:
            db[key] = value
            return True
        except:
            # тут желательно как-то обработать ситуацию
            return False


# Создание ключа для записи и чтения
def make_key(chatid, keyid):
    res = str(chatid) + '__' + str(keyid)
    return res
```

Файл «config.py»:

```
from enum import Enum



# Токент бота

TOKEN = '5082279521:AAGZCqpuWDu2a0F1XMTVCcmdzSRwvaL9_1Y'



# Файл базы данных Vedis

db_file = "db.vdb"



# Ключ записи в БД для текущего состояния

CURRENT_STATE = "CURRENT_STATE"



SENTENCE = "SENTENCE"



# Состояния автомата
```

```
class States(Enum):

    STATE_START = "STATE_START"  # Начало нового диалога

    STATE_FIRST = "STATE_FIRST"

    STATE_SECOND = "STATE_SECOND"

    STATE_THIRD = "STATE_THIRD"

    STATE_OPERATION = "STATE_OPERATION"
```

Файл «main.py»:

```
import os

import telebot
from telebot import types

import config
from dbworker import *
import random

# Создание бота
bot = telebot.TeleBot(config.TOKEN)

# путь к текущему каталогу
cur_path = os.path.dirname(os.path.abspath(__file__))

def find_path(a, b, c):
    if a=="Длинные": res='0'
    else: res='1'
    if b=="Яркие": res+='0'
    else: res+='1'
    if c=="С дизайном": res+='0'
    else: res+='1'
    path = "C:/Users/ASUS/PycharmProjects/Lab5/%s"%res
    return path


@bot.message_handler(commands=["start"])
def start(message):
    markup = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    ans = "Даа"
    markup.add(ans)
    bot.send_message(message.chat.id,"Йоу! Я бот by d_romashhh, давай
посоветую тебе Ты готова?",reply_markup=markup)
    set(make_key(message.chat.id,
config.CURRENT_STATE),config.States.STATE_FIRST.value)


@bot.message_handler(commands=["restart"])
def start(message):
    bot.send_message(message.chat.id,"Произошел перезапуск")
    set(make_key(message.chat.id,
config.CURRENT_STATE),config.States.STATE_FIRST.value)

@bot.message_handler(func=lambda
message:get(make_key(message.chat.id,config.CURRENT_STATE))==config.States.ST
ATE_FIRST.value)
def state_first(message): #message- хранит информацию введенную пользователем
    text=message.text
    markup=types.ReplyKeyboardMarkup(row_width=2,resize_keyboard=True) #то
что выводятся кнопки
    but1=types.KeyboardButton("Короткие")
```

```
    but2 = types.KeyboardButton("Длинные")
    markup.add(but1,but2)
    bot.send_message(message.chat.id,"Для начала выбери длину
ногтей💅",reply_markup=markup)
    #markup = types.ReplyKeyboardRemove(selective=False)
    #bot.send_message(message.chat.id, "Первый выводимый текст",
reply_markup=markup)
    set(make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SECOND.value)#переход в следующее состояние по ключу

@bot.message_handler(func=lambda
message:get(make_key(message.chat.id,config.CURRENT_STATE))==config.States.ST
ATE_SECOND.value)
def state_second(message): #message- хранит информацию введенную
пользователем
    text=message.text
    set(make_key(message.chat.id, config.States.STATE_FIRST), text)
    markup=types.ReplyKeyboardMarkup(row_width=2,resize_keyboard=True) #то
что выводятся кнопки
    but1=types.KeyboardButton("Нюдовые")
    but2 = types.KeyboardButton("Яркие")
    markup.add(but1,but2)
    bot.send_message(message.chat.id,"Теперь выбери
стиль😇",reply_markup=markup)
    #markup = types.ReplyKeyboardRemove(selective=False)
    #bot.send_message(message.chat.id, "Пр", reply_markup=markup)
    set(make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_THIRD.value)


@bot.message_handler(func=lambda
message:get(make_key(message.chat.id,config.CURRENT_STATE))==config.States.ST
ATE_THIRD.value)
def state_third(message): #message- хранит информацию введенную пользователем
    text=message.text
    set(make_key(message.chat.id, config.States.STATE_SECOND), text)
    markup=types.ReplyKeyboardMarkup(row_width=2,resize_keyboard=True) #то
что выводятся кнопки
    but1=types.KeyboardButton("С дизайном")
    but2 = types.KeyboardButton("Без дизайна")
    markup.add(but1,but2)
    bot.send_message(message.chat.id,"И последний
вопрос♡",reply_markup=markup)
    set(make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_OPERATION.value)


@bot.message_handler(func=lambda
message:get(make_key(message.chat.id,config.CURRENT_STATE))==config.States.ST
ATE_OPERATION.value)
def state_operation(message): #message- хранит информацию введенную
пользователем
    design=message.text
    length=get(make_key(message.chat.id,config.States.STATE_FIRST))
    bright = get(make_key(message.chat.id, config.States.STATE_SECOND))
    path=find_path(length, bright, design)
    photo = open(path+"/" + random.choice(os.listdir(path)), 'rb')
    markup = types.ReplyKeyboardRemove(selective=False)
    bot.send_photo(message.chat.id, photo)
    bot.send_message(message.chat.id, "Надеюсь тебе
понравилось🏵",reply_markup=markup)
    set(make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_FIRST.value)
```

```
    markup = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    ans = "Даа"
    markup.add(ans)
    bot.send_message(message.chat.id, "Йоу! Я бот by d_romashhh, давай
посоветую тебе Ты готова?", reply_markup=markup)


if __name__ == '__main__':
    bot.infinity_polling()
```

Файл «tests.py»:

```
import unittest
from main import *

class MyTestCase(unittest.TestCase):
    def test_1(self):
        self.assertEqual(find_path("Длинные","Яркие", "С
дизайном"),"C:/Users/ASUS/PycharmProjects/Lab5/000")  # add assertion here
    def test_2(self):
        self.assertEqual(find_path("Короткие", "Нюдовые", "С дизайном"),
"C:/Users/ASUS/PycharmProjects/Lab5/110")

if __name__ == '__main__':
    unittest.main()
```

Файл «stepsBDD.py»:

```
from behave import given, when, then
from main import find_path

@given('choises "{a}", "{b}", "{c}"')
def given_c(context, a, b, c):
    context.a = a
    context.b = b
    context.c = c

@when("Something")
def calculation(context):
    context.result = find_path(context.a, context.b, context.c)

@then('Result "{result}"')
def get_result(context, result):
    assert context.result == result
```

Файл «featureBDD.feature»:

```
Feature: Nails bot

  Scenario: first test
    Given choises "Длинные", "Яркие", "С дизайном"
    When Something
    Then Result "C:/Users/ASUS/PycharmProjects/Lab5/000"
  Scenario: second test
    Given choises "Короткие", "Нюдовые", "С дизайном"
    When Something
    Then Result "C:/Users/ASUS/PycharmProjects/Lab5/110"
```

Результат выполнения программы:

```
PS C:\Users\ASUS\PycharmProjects\Lab5> behave
Feature: Nails bot # featureBDD.feature:2

  Scenario: first test                                    # featureBDD.feature:4
    Given choises "Длинные", "Яркие", "С дизайном"        # steps/stepsBDD.py:4
    When Something                                        # steps/stepsBDD.py:10
    Then Result "C:/Users/ASUS/PycharmProjects/Lab5/000" # steps/stepsBDD.py:14

  Scenario: second test                                   # featureBDD.feature:8
    Given choises "Короткие", "Нюдовые", "С дизайном"     # steps/stepsBDD.py:4
    When Something                                        # steps/stepsBDD.py:10
    Then Result "C:/Users/ASUS/PycharmProjects/Lab5/110" # steps/stepsBDD.py:14

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.004s
```

```
Ran 2 tests in 0.005s

OK

Process finished with exit code 0
```