**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы программирования»

Отчет по домашнему заданию

Выполнила: Кичикова Александра

Группа: ИУ5- 32Б

Москва, 2021 г.

Задание:

1. Модифицируйте код лабораторной работы №6 таким образом, чтобы он был пригоден для модульного тестирования.
2. Используя материалы лабораторной работы №4 создайте модульные тесты с применением TDD - фреймворка (2 теста) и BDD - фреймворка (2 теста).

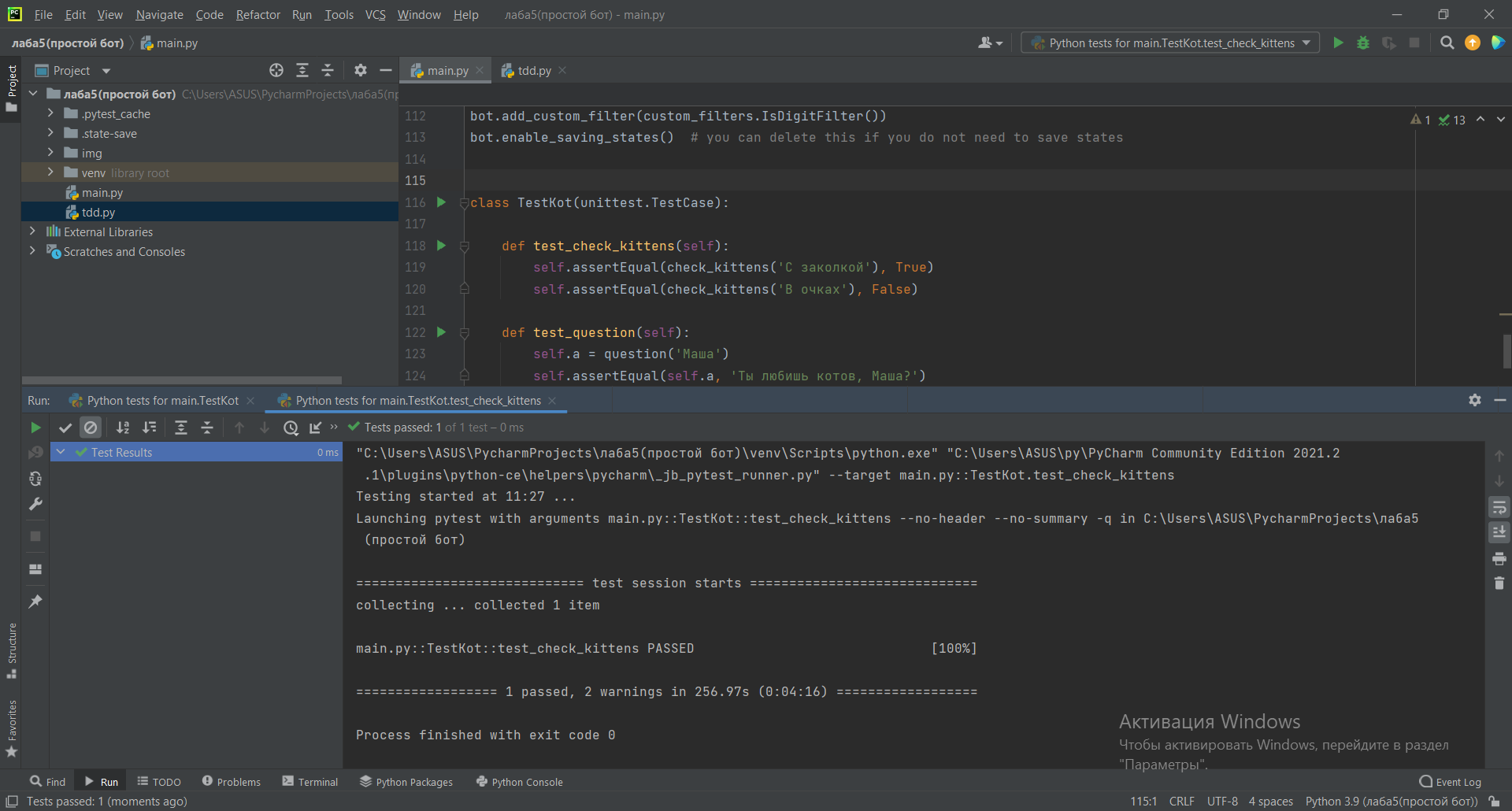
Текст программы:

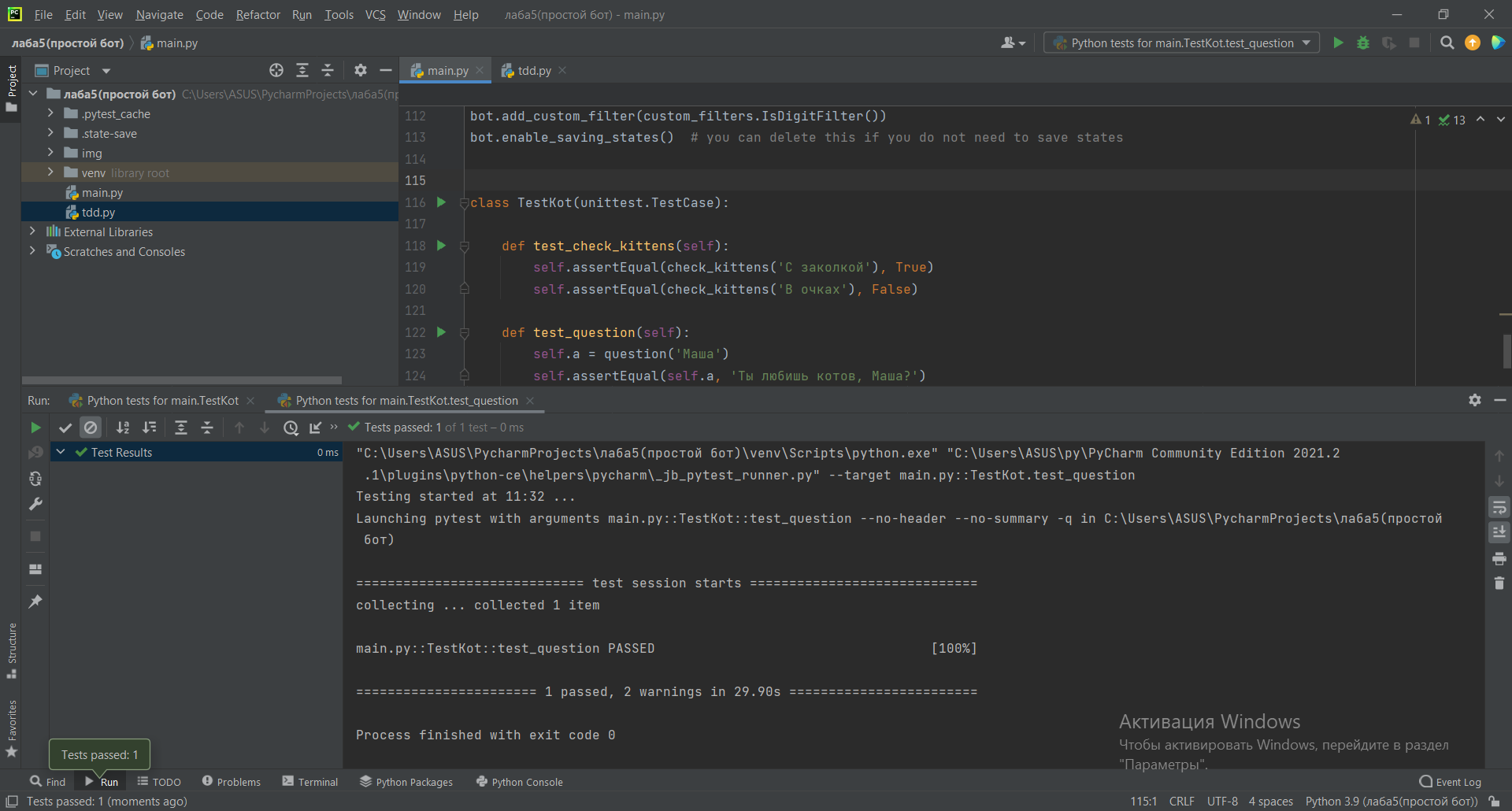
**Main.py**

import telebot  
import os  
from telebot import types  
from telebot import custom\_filters  
import asyncio  
  
cur\_path = os.path.dirname(os.path.abspath(\_\_file\_\_))  
  
TOKEN = '2128753305:AAGkplNGruiJZ74\_B0p1f4wCHFS1NykUuL4'  
bot = telebot.TeleBot(TOKEN)  
kittens = ['С заколкой', 'В галстуке-бабочке', 'Не котёнок', 'В шапочке']  
  
  
def question(name):  
 a = 'Ты любишь котов, ' + name + '?'  
 return a  
  
mes\_hairclip = 'С заколкой'  
mes\_bowtie = 'В галстуке-бабочке'  
mes\_notkitten = 'Не котёнок'  
mes\_hat = 'В шапочке'  
  
  
class MyStates:  
 name = 1  
 doyou = 2  
 wannasee = 3  
 picture = 4  
  
@bot.message\_handler(commands=['start'])  
def start\_ex(message):  
 bot.set\_state(message.from\_user.id, MyStates.name)  
 bot.send\_message(message.chat.id, 'Привет! Как тебя зовут?')  
  
  
@bot.message\_handler(state=MyStates.name)  
def name\_get(message):  
 name = message.text  
 markup = types.ReplyKeyboardMarkup(row\_width=2)  
 markup.add('Да', 'Нет')  
 bot.send\_message(message.chat.id, question(name), reply\_markup=markup)  
 bot.set\_state(message.from\_user.id, MyStates.doyou)  
  
  
async def love(message):  
 bot.send\_message(message.chat.id, 'Я могу подождать, пока полюбите :)')  
 await asyncio.sleep(3)  
 bot.send\_message(message.chat.id, 'Ну как? Полюбили?')  
  
  
@bot.message\_handler(state=MyStates.doyou)  
def ask\_about\_cats(message):  
 if message.text == 'Нет':  
 asyncio.run(love(message))  
 else:  
 markup = types.ReplyKeyboardMarkup(row\_width=2)  
 markup.add('Хочу', 'Не хочу')  
 bot.send\_message(message.chat.id, 'Я тоже!! Хочешь пришлю картинку?', reply\_markup=markup)  
 bot.set\_state(message.from\_user.id, MyStates.wannasee)  
  
  
async def pic(message):  
 bot.send\_message(message.chat.id, 'Обращайся, если захочешь :)')  
 await asyncio.sleep(3)  
 bot.send\_message(message.chat.id, 'А сейчас хочешь картиночку?')  
  
  
@bot.message\_handler(state=MyStates.wannasee)  
def ask\_about\_a\_picture(message):  
 if message.text == 'Не хочу':  
 asyncio.run(pic(message))  
 else:  
 markup = types.ReplyKeyboardMarkup(row\_width=2)  
 itembtn1 = types.KeyboardButton(mes\_hairclip)  
 itembtn2 = types.KeyboardButton(mes\_bowtie)  
 itembtn3 = types.KeyboardButton(mes\_notkitten)  
 itembtn4 = types.KeyboardButton(mes\_hat)  
 markup.add(itembtn1, itembtn2, itembtn3, itembtn4)  
 bot.send\_message(message.chat.id, 'Чудесно! Какого котенка ты хочешь увидеть?', reply\_markup=markup)  
 bot.set\_state(message.from\_user.id, MyStates.picture)  
  
def check\_kittens(text):  
 return text in kittens  
  
@bot.message\_handler(state=MyStates.picture)  
def ask\_what\_kitten(message):  
 chat\_id = message.chat.id  
 text = message.text  
 if check\_kittens(text):  
 if text == mes\_hairclip:  
 img = open(os.path.join(cur\_path, 'img\hairclip.jpg'), 'rb')  
 bot.send\_photo(chat\_id, img)  
 elif text == mes\_bowtie:  
 img = open(os.path.join(cur\_path, 'img/bowtie.jpg'), 'rb')  
 bot.send\_photo(chat\_id, img)  
 elif text == mes\_notkitten:  
 img = open(os.path.join(cur\_path, 'img/notkitten.jpg'), 'rb')  
 bot.send\_photo(chat\_id, img)  
 elif text == mes\_hat:  
 img = open(os.path.join(cur\_path, 'img/hat.jpg'), 'rb')  
 bot.send\_photo(chat\_id, img)  
 else:  
 bot.send\_message(message.chat.id, 'Такого предложить не могу :(')  
  
  
bot.add\_custom\_filter(custom\_filters.StateFilter(bot))  
bot.add\_custom\_filter(custom\_filters.IsDigitFilter())  
bot.enable\_saving\_states()

class TestKot(unittest.TestCase):  
  
 def test\_check\_kittens(self):  
 self.assertEqual(check\_kittens('С заколкой'), True)  
 self.assertEqual(check\_kittens('В очках'), False)  
  
 def test\_question(self):  
 self.a = question('Маша')  
 self.assertEqual(self.a, 'Ты любишь котов, Маша?')  
  
bot.infinity\_polling(skip\_pending=True)

Результаты тестирования:





Экранные формы примеров выполнения программы:

