

**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
Ковыршин Павел

Подпись и дата:

Проверил:

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г.

Постановка задачи.

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.

values.py

```
from enum import Enum

class state(Enum):
    DEFAULT = 1
    PICTURE = 2
    PLOT = 3

current_state = state.DEFAULT
prog_path = 'C:/Users/pahan/python_prog/dz'
fig_filename = 'fig.png'
kovyrshintoken = '5077381184:AAE5d6etfiLN8K7SbPwabfuGAXYG6xuPl4'
commands = ['start', 'help', 'options', 'pics']
pic1_url =
'https://ichef.bbci.co.uk/news/640/cpsprodpb/14236/production/_104368428_gettyimages-543560762.jpg'
pic2_url = 'https://www.belnovosti.by/sites/default/files/2020-02/ezh_0.jpg'
first_pic = 'первая картинка'
second_pic = 'вторая картинка'
eitherpic_r = ('(^' + first_pic + ')|(^' + second_pic + ')')
pics_r = r'^картинк[аиу]'
plot_r = r'(((по)?стро[ий]).*(графи[кч]))|(((графи[кч]).*((по)?стро[ий])))'
plotted_func_r = r'^.х.'
```

pics.py

```
from urllib import request

def get_pic(url):
    return request.urlopen(url)
```

plotter.py

```
import matplotlib
import values
import numpy as np
from matplotlib import pyplot as plt
matplotlib.use('Agg')
from math import *
```

```

def str_to_func(str):
    try:
        return eval('lambda x: ' + str.lower())
    except SyntaxError:
        return lambda x: 0

def plot(str):
    fig, ax = plt.subplots()
    f = str_to_func(str)
    x = np.linspace(-10, 10, 10000)
    fx = np.array([])
    x_good = np.array([])
    for x0 in x:
        try:
            if abs(f(x0)) > 200:
                raise ValueError
            fx = np.append(fx, f(x0))
            x_good = np.append(x_good, x0)
        except ValueError:
            pass
        except OverflowError:
            pass
        except ZeroDivisionError:
            pass
    ax.plot(x_good, fx)
    plot_file = values.prog_path + '/' + values.fig_filename
    fig.savefig(plot_file)
    return plot_file

if __name__ == '__main__':
    plot('exp(1/x)')

```

custom_filter.py

```

import telebot
import values

class CurrentState(telebot.AdvancedCustomFilter):
    key = 'current_state'
    @staticmethod
    def check(message, current_state):
        return values.current_state in current_state

```

bot.py

```

import telebot
from telebot.types import ReplyKeyboardRemove
import os
from custom_filter import CurrentState
from values import state
import values

```

```

from pics import get_pic
import plotter

kovyrshinbot = telebot.TeleBot('5077381184:AAE5d6etfiLN8K7SbPwabfuGAXYG6xuPla4')

@kovyrshinbot.message_handler(commands=['state'])
def start(message):
    kovyrshinbot.send_message(message.chat.id, text= str(values.current_state))

@kovyrshinbot.message_handler(commands=['start'])
def start(message):
    values.current_state = state.DEFAULT
    remove_markup = ReplyKeyboardRemove()
    kovyrshinbot.send_message(message.chat.id, text= 'Я завелся, чтобы строить графики или
делать еще что-то', reply_markup= remove_markup)

@kovyrshinbot.message_handler(commands=['help'])
def help(message):
    values.current_state = state.DEFAULT
    remove_markup = ReplyKeyboardRemove()
    kovyrshinbot.send_message(message.chat.id, text= '/start - старт\n/options -
варианты\n/pics - картинки\n/help - помощь', reply_markup= remove_markup)

@kovyrshinbot.message_handler(commands=['options'])
def options(message):
    values.current_state = state.DEFAULT
    markup = telebot.types.ReplyKeyboardMarkup(resize_keyboard= True)
    plot_button = telebot.types.KeyboardButton('Построй график')
    extra_button = telebot.types.KeyboardButton('Картинки')
    markup.add(plot_button)
    markup.add(extra_button)
    kovyrshinbot.send_message(message.chat.id, text='Ну вот варианты', reply_markup=
markup)

@kovyrshinbot.message_handler(commands=['pics'])
def pics_command(message):
    pics(message)

@kovyrshinbot.message_handler(current_state= [state.DEFAULT], content_types= ['text'],
regex= values.pics_r)
def pics_text(message):
    pics(message)

def pics(message):
    values.current_state = state.PICTURE
    markup = telebot.types.ReplyKeyboardMarkup(resize_keyboard= True)
    pic1_button = telebot.types.KeyboardButton('Первая картинка')
    pic2_button = telebot.types.KeyboardButton('Вторая картинка')
    markup.add(pic1_button)
    markup.add(pic2_button)
    kovyrshinbot.send_message(message.chat.id, text='Какую тебе картинку', reply_markup=
markup)

```

```

@kovyrshinbot.message_handler(current_state= [state.PICTURE], content_types= ['text'])
def send_pic(message):
    if (message.text.lower() == values.first_pic):
        pic = get_pic(values.pic1_url)
    elif (message.text.lower() == values.second_pic):
        pic = get_pic(values.pic2_url)
    else:
        remove_markup = ReplyKeyboardRemove()
        values.current_state = state.DEFAULT
        kovyrshinbot.send_message(message.chat.id, text= 'Че?', reply_markup=
remove_markup)
        remove_markup = ReplyKeyboardRemove()
        values.current_state = state.DEFAULT
        kovyrshinbot.send_photo(message.chat.id, photo= pic, reply_markup= remove_markup)

@kovyrshinbot.message_handler(content_types= ['text'], regexp= values.plot_r)
def plot(message):
    values.current_state = state.PLOT
    remove_markup = ReplyKeyboardRemove()
    kovyrshinbot.send_message(message.chat.id, text= 'Введи функцию с переменной x в
синтаксисе python.\nКое-как построю график в пределах \n[-10, 10] или меньше',
reply_markup= remove_markup)

@kovyrshinbot.message_handler(current_state= [state.PLOT], content_types= ['text'])
def plot_picture(message):
    values.current_state = state.DEFAULT
    remove_markup = ReplyKeyboardRemove()
    plot_path = plotter.plot(message.text)
    plot_img = open(plot_path, 'rb')
    kovyrshinbot.send_photo(message.chat.id, photo= plot_img, reply_markup= remove_markup)
    plot_img.close()
    os.remove(plot_path)

kovyrshinbot.add_custom_filter(CurrentState())
kovyrshinbot.infinity_polling()

```

testTDD.py

```

from custom_filter import CurrentState
import values
from values import state
import plotter
import unittest
import os

class bot_test(unittest.TestCase):
    def test_statecheck_default(self):
        values.current_state = state.DEFAULT
        self.assertTrue(CurrentState.check(None, [state.DEFAULT]))
    def test_plot_created(self):

```

```

        plot_path = plotter.plot('x')
        self.assertTrue(os.path.exists(plot_path))
        os.remove(plot_path)

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

```

steps/stepsBDD.py

```

from behave import given, when, then
from custom_filter import CurrentState
from values import state
import values
import plotter
import os

@given("required states are {required_states}, current state is {current_state}")
def given_c(context, required_states, current_state):
    values.current_state = state[current_state]
    context.required_states = list(map(state.__getitem__, required_states.split(', ')))

@when("checking for state")
def calculation(context):
    context.result = CurrentState.check(None, context.required_states)

@then("current state being in required is {result}")
def get_result(context, result):
    result = bool(result)
    assert context.result == result

@given("the plotted function is '{fstr}'")
def given_c(context, fstr):
    context.plot_path = plotter.plot(fstr)

@when("plotting and saving the file")
def calculation(context):
    context.result = os.path.exists(context.plot_path)

@then("it is {result} that the file is there")
def get_result(context, result):
    result = bool(result)
    assert context.result == result
    if result:
        os.remove(context.plot_path)

```

featureBDD.feature

Feature: Testing bot functionality

Scenario: Checking if current state is in the required state list

Given required states are DEFAULT, current state is DEFAULT

When checking for state

```
The current state being in required is True
```

```
Scenario: Checking if the plot file is created
```

```
Given the plotted function is 'abs(x) if x < 0 else log(x)'
```

```
When plotting and saving the file
```

```
Then it is True that the file is there
```

Результат выполнения программы.

testTDD.py

```
C:\Users\pahan\python_prog\dz>python testTDD.py
```

```
..
```

```
-----  
Ran 2 tests in 0.304s
```

```
OK
```

behave

```
C:\Users\pahan\python_prog\dz>behave
```

```
Feature: Testing bot functionality # featureBDD.feature:1
```

```
Scenario: Checking if current state is in the required state list # featureBDD.feature:3  
  Given required states are DEFAULT, current state is DEFAULT # steps/stepsBDD.py:8  
  When checking for state # steps/stepsBDD.py:13  
  Then current state being in required is True # steps/stepsBDD.py:17
```

```
Scenario: Checking if the plot file is created # featureBDD.feature:8  
  Given the plotted function is 'abs(x) if x < 0 else log(x)' # steps/stepsBDD.py:22  
  When plotting and saving the file # steps/stepsBDD.py:26  
  Then it is True that the file is there # steps/stepsBDD.py:30
```

```
1 feature passed, 0 failed, 0 skipped
```

```
2 scenarios passed, 0 failed, 0 skipped
```

```
6 steps passed, 0 failed, 0 skipped, 0 undefined
```

```
Took 0m0.288s
```