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»

Отчет по домашнему заданию «Основные конструкции языка Python»

Выполнил:
студент группы ИУ5-32Б
Зайцев А.Д.

Москва, 2021 г.

Описание задания:

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:
Файл asyncbot3.py

```
from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher
from aiogram.dispatcher.filters.state import StatesGroup,State
from aiogram.dispatcher.filters import Command
from aiogram.dispatcher.storage import RESULT, FSMContext
from aiogram.utils import executor
from config import TOKEN
from aiogram.types import ReplyKeyboardRemove,ReplyKeyboardMarkup,
KeyboardButton, InlineKeyboardMarkup, InlineKeyboardButton,Message

from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.contrib.middlewares.logging import LoggingMiddleware


bot = Bot(token=TOKEN)
dp = Dispatcher(bot, storage=MemoryStorage())
dp.middleware.setup(LoggingMiddleware())


class Test(StatesGroup):
    Q0 = State()
    Q1 = State()
    Q2 = State()
    Q3 = State()

first_course={
    1 : "Уха",
    2 : "Борщ",
    3 : "Солянка"
}
main_course={
    1 : "Спагетти карбонара",
    2 : "Котлета от Шефа",
    3 : "Наггетсы в сычуанском соусе"
}
dessert={
    1 : "Шарик сливочного мороженого",
    2 : "Чизкейк",
    3 : "Макфлури"
}
```

```
first_course_price = {
    "Уха":50,
    "Борщ": 75,
    "Солянка": 70
}
main_course_price = {
    "Спагетти карбонара": 150,
    "Котлета от Шефа": 175,
    "Наггетсы в сычуанском соусе": 200
}
dessert_price={
    "Шарик сливочного мороженого" : 50,
    "Чизкейк" : 180,
    "Макфлури" : 99
}
def
make_check(first_table,first_table_price,second_table,second_table_price,third_ta
ble,third_table_price,first_answer,second_answer,third_answer):
    return "Ваш заказ:\nПервое блюдо:\n{} - {}\nОсновное блюдо:\n{} -
{}\nДесерт:\n{} -
{}\n".format(first_table[first_answer],first_table_price[first_table[first_answer
]],second_table[second_answer],second_table_price[second_table[second_answer]],th
ird_table[third_answer],third_table_price[third_table[third_answer]])
def
summary(first_table,first_table_price,second_table,second_table_price,third_table
,third_table_price,first_answer,second_answer,third_answer):
    a = first_table_price[first_table[first_answer]]
    b = second_table_price[second_table[second_answer]]
    c = third_table_price[third_table[third_answer]]
    return "Итоговая сумма = " + str(a+b+c)+ "\n"
def
average_price(first_table,first_table_price,second_table,second_table_price,third
_table,third_table_price,first_answer,second_answer,third_answer):
    a = first_table_price[first_table[first_answer]]
    b = second_table_price[second_table[second_answer]]
    c = third_table_price[third_table[third_answer]]
    return "Средняя стоимость одной позиции = " + str(int((a+b+c)/3))

@dp.message_handler(state="*", commands=['start'])
async def starting_process1(message: types.Message):
    await bot.send_message(message.from_user.id,"Привет\nты можешь сделать заказ
из 3 блюд\n1)Первое блюдо\n2)Второе блюдо\n3)Десерт\nЧтобы начать формировать
заказ напиши /form\nЕсли хочешь начать формировать заказ заново, напиши /clear")
    await Test.Q0.set()

@dp.message_handler(state="*", commands=['clear'])
async def starting_process2(message: types.Message,state: FSMContext):
    await state.finish()
    await Test.Q0.set()
```

```python
#вызов команды /form в процессе заказа
@dp.message_handler(state=Test.Q1, commands=['form'])
async def refuse_command1(message: types.Message ):
    await bot.send_message(message.from_user.id,"прежде чем формировать новый
заказ,завершите предыдущий либо напишите /clear")
@dp.message_handler(state=Test.Q2, commands=['form'])
async def refuse_command2(message: types.Message ):
    await bot.send_message(message.from_user.id,"прежде чем формировать новый
заказ,завершите предыдущий либо напишите /clear")
@dp.message_handler(state=Test.Q3, commands=['form'])
async def refuse_command3(message: types.Message ):
    await bot.send_message(message.from_user.id,"прежде чем формировать новый
заказ,завершите предыдущий либо напишите /clear")


@dp.message_handler(state=Test.Q0, commands=['form'])
async def starting_forming(message: types.Message,state: FSMContext):
    await bot.send_message(message.from_user.id, "Выбери первое блюдо\n1)Уха -
50р.\n2)Борщ - 75р.\n3)Солянка - 70р.")
    await Test.Q1.set()

@dp.message_handler(state=Test.Q1)
async def first_choosing(message: types.Message,state: FSMContext):

    answer = int(message.text)
    if (answer != 1 and answer !=2 and answer !=3):
        return await bot.send_message(message.from_user.id,"такого варианта
ответа нет выбери еще раз!!!!!!!!")
    await state.update_data(q1 = answer)
    await bot.send_message(message.from_user.id,"Выбери второе блюдо\n1)спагетти
карбонара - 150р.\n2)котлета от шефа - 175р.\n3)наггетсы в сычуанском соусе -
200р.")
    await Test.Q2.set()

@dp.message_handler(state=Test.Q2)
async def second_choosing(message: types.Message,state: FSMContext):
    answer = int(message.text)
    if (answer != 1 and answer !=2 and answer !=3):
        return await bot.send_message(message.from_user.id,"такого варианта
ответа нет выбери еще раз!!!!!!!!")
    await state.update_data(q2 = answer)
    await bot.send_message(message.from_user.id, "Выбери десерт\n1)шарик
сливочного мороженого - 50р.\n2)чизкейк - 180р.\n3)макфлури - 99р.")
    await Test.Q3.set()



@dp.message_handler(state=Test.Q3)
async def third_choosing(message: types.Message,state: FSMContext):
    answer = int(message.text)
    if (answer != 1 and answer !=2 and answer !=3):
```

```
        return await bot.send_message(message.from_user.id,"такого варианта
ответа нет выбери еще раз!!!!!!!!")
    await state.update_data(q3 = answer)
    data = await state.get_data()
    check =
make_check(first_course,first_course_price,main_course,main_course_price,dessert,
dessert_price,data.get("q1"),data.get("q2"),data.get("q3"))
    sumcheck=summary(first_course,first_course_price,main_course,main_course_pric
e,dessert,dessert_price,data.get("q1"),data.get("q2"),data.get("q3"))
    aver=average_price(first_course,first_course_price,main_course,main_course_pr
ice,dessert,dessert_price,data.get("q1"),data.get("q2"),data.get("q3"))
    #await bot.send_message(message.from_user.id,"Ваш заказ:\nПервое блюдо:\n{} -
{}\nОсновное блюдо:\n{} - {}\nДесерт:\n{} -
{}".format(first_course[data.get("q1")],first_course_price[first_course[data.get(
"q1")]],main_course[data.get("q2")],main_course_price[main_course[data.get("q2")]
],dessert[data.get("q3")],dessert_price[dessert[data.get("q3")]]))
    await bot.send_message(message.from_user.id,check+sumcheck+aver)
    await Test.Q0.set()


async def shutdown(dispatcher: Dispatcher):
    await dispatcher.storage.close()
    await dispatcher.storage.wait_closed()

if __name__ == '__main__':
    executor.start_polling(dp, on_shutdown=shutdown)
```

Файл config.py:

```
TOKEN="5086492621:AAEhnZviAv4rJ_gQ64dnPluMfrgWh1l13FU"
```

Файл test_asyncbot.py:

```
import unittest
from asyncbot3 import
summary,average_price,first_course,first_course_price,main_course,main_course_pri
ce,dessert,dessert_price




class Bot_test(unittest.TestCase):
    def test_summary(self):
        ans1 = 1
        ans2 = 2
        ans3 = 3
        res =
summary(first_course,first_course_price,main_course,main_course_price,dessert,des
sert_price,ans1,ans2,ans3)
        self.assertEqual("Итоговая сумма = 324\n",res)
    def test_average_price(self):
```

```
        ans1 = 1
        ans2 = 2
        ans3 = 3
        res =
average_price(first_course,first_course_price,main_course,main_course_price,desse
rt,dessert_price,ans1,ans2,ans3)
        self.assertEqual("Средняя стоимость одной позиции = 108.0",res)




if __name__ == "__name__":
    unittest.main()
```

Файл test_average_bot.feature:

```
Feature: Test average
    Scenario: test average for making check with 1 2 3
        Given ordering1 food with answers in bot first course - 1 main course - 2
dessert - 3
        When we form average price of position of check
        Then average price should 108
```

Файл test_sum_bot.feature:

```
Feature: Test summary
    Scenario: test summary for making check with 1 2 3
        Given ordering food with answers in bot first course - 1 main course - 2
dessert - 3
        When we form summary of check
        Then check should be with correct price 324
    Scenario: test summary for making check with 1 3 3
        Given ordering food with answers in bot first course - 1 main course - 3
dessert - 3
        When we form summary of check
        Then check should be with correct price 349
    Scenario: test summary for making check with 2 2 2
        Given ordering food with answers in bot first course - 2 main course - 2
dessert - 2
        When we form summary of check
        Then check should be with correct price 430
```

Файл test_average.py:

```
from behave import Given,When,Then
from asyncbot3 import
summary,average_price,first_course,first_course_price,main_course,main_course_pri
ce,dessert,dessert_price



@Given("ordering1 food with answers in bot first course - {a} main course - {b}
dessert - {c}")
def given_answers(context,a,b,c):
    context.ans1=int(a)
```

```
    context.ans2=int(b)
    context.ans3=int(c)

@When("we form average price of position of check")
def make_summary(context):
    res =
average_price(first_course,first_course_price,main_course,main_course_price,desse
rt,dessert_price,context.ans1,context.ans2,context.ans3)
    context.result=res

@Then("average price should {d}")
def compare_results(context,d):
    corres= "Средняя стоимость одной позиции = " + str(d)
    assert(context.result == corres )
```

Файл test_summary.py:

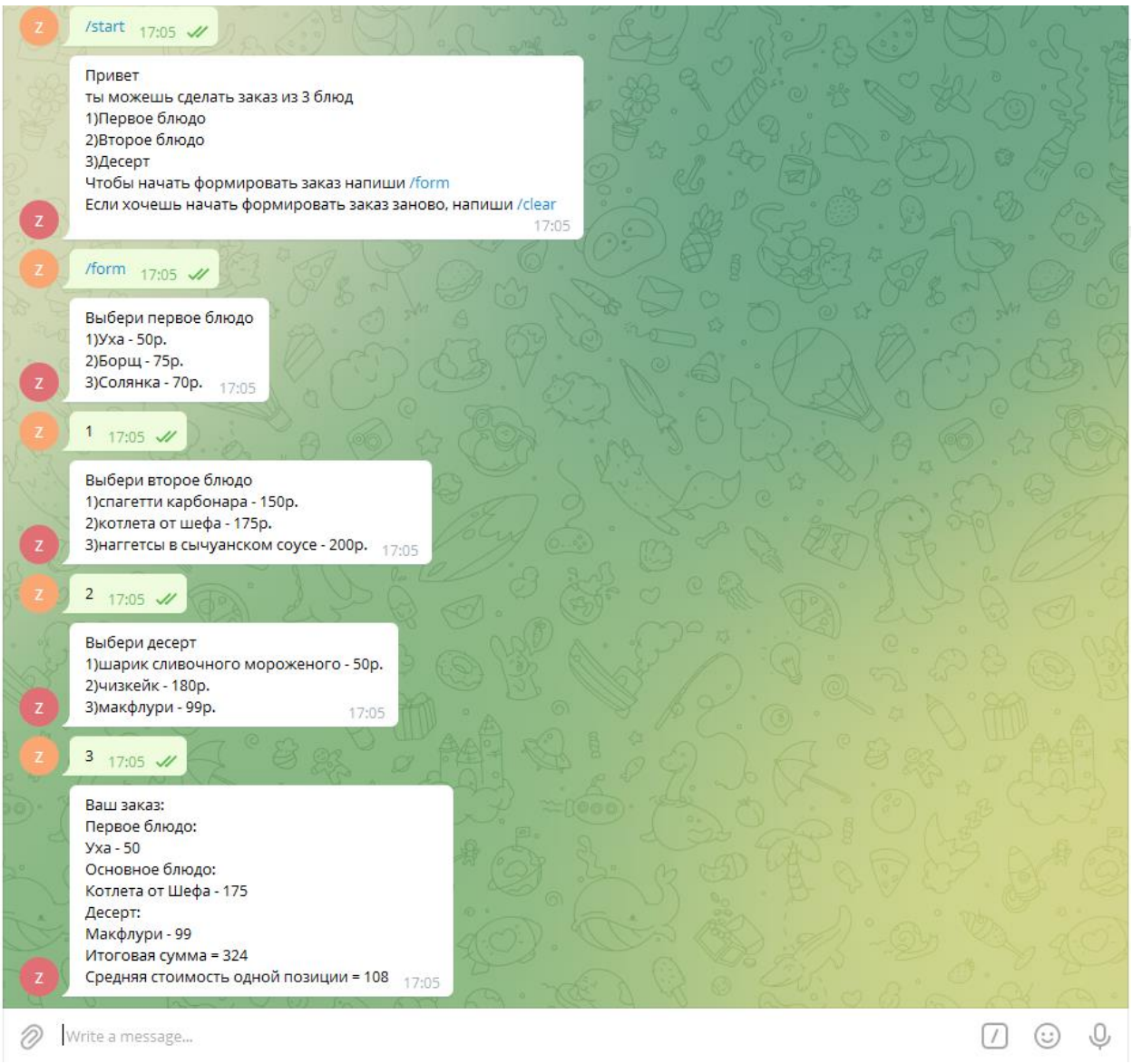
```
from behave import Given,When,Then
from asyncbot3 import
summary,average_price,first_course,first_course_price,main_course,main_course_pri
ce,dessert,dessert_price

@Given("ordering food with answers in bot first course - {a} main course - {b}
dessert - {c}")
def given_answers(context,a,b,c):
    context.ans1=int(a)
    context.ans2=int(b)
    context.ans3=int(c)

@When("we form summary of check")
def make_summary(context):
    res =
summary(first_course,first_course_price,main_course,main_course_price,dessert,des
sert_price,context.ans1,context.ans2,context.ans3)
    context.result=res

@Then("check should be with correct price {d}")
def compare_results(context,d):
    corres= "Итоговая сумма = " + str(d)+ "\n"
    assert(context.result == corres )
```

Результат выполнения программы:



Результат работы тестов:

TDD:

test_asyncbot3.py

```
PS C:\Users\Scare\Documents\BKIT1\DZ> python -m unittest test_asyncbot.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
PS C:\Users\Scare\Documents\BKIT1\DZ>
```

BDD:

feature: test_average_bot.feature, test_sum_bot.feature

steps: test_average.py, test_summary.py

```
PS C:\Users\Scare\Documents\BKIT1\DZ> behave
Feature: Test average # feature/test_average_bot.feature:1

  Scenario: test average for making check with 1 2 3                                  # feature/test_average_bot.feature:2
    Given ordering1 food with answers in bot first course - 1 main course - 2 dessert - 3 # steps/test_average.py:6
    When we form average price of position of check                                   # steps/test_average.py:12
    Then average price should 108                                                     # steps/test_average.py:17

Feature: Test summary # feature/test_sum_bot.feature:1

  Scenario: test summary for making check with 1 2 3                                 # feature/test_sum_bot.feature:2
    Given ordering food with answers in bot first course - 1 main course - 2 dessert - 3 # steps/test_summary.py:4
    When we form summary of check                                                    # steps/test_summary.py:10
    Then check should be with correct price 324                                      # steps/test_summary.py:15

  Scenario: test summary for making check with 1 3 3                                 # feature/test_sum_bot.feature:6
    Given ordering food with answers in bot first course - 1 main course - 3 dessert - 3 # steps/test_summary.py:4
    When we form summary of check                                                    # steps/test_summary.py:10
    Then check should be with correct price 349                                      # steps/test_summary.py:15

  Scenario: test summary for making check with 2 2 2                                 # feature/test_sum_bot.feature:10
    Given ordering food with answers in bot first course - 2 main course - 2 dessert - 2 # steps/test_summary.py:4
    When we form summary of check                                                    # steps/test_summary.py:10
    Then check should be with correct price 430                                      # steps/test_summary.py:15

2 features passed, 0 failed, 0 skipped
4 scenarios passed, 0 failed, 0 skipped
12 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.004s
PS C:\Users\Scare\Documents\BKIT1\DZ>
```