**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет ИУ

Кафедра ИУ5

Курс «Основы информатики»

Отчет по лабораторной работе №\_5\_

«Модульное тестирование в Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-23Б: |  | преподаватель каф. \_\_\_\_ |
| Селиханович Д.С. |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

**Постановка задачи**

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:

* TDD - фреймворк (не менее 3 тестов).
* BDD - фреймворк (не менее 3 тестов).
* Создание Mock-объектов (необязательное дополнительное задание).

**Текст программы**

**\lab\_03­\_04\process\_data.py**

import json

import sys

from print\_result import print\_result

from gen\_random import gen\_random

from cm\_timer import cm\_timer\_1

# Сделаем другие необходимые импорты

path = "C:\\Users\\user\\Desktop\\prog\\Python\\lab\_03\_04\\data\_light.json"

# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске сценария

with open(path, 'r', encoding='utf-8') as f:

    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`

# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку

# В реализации функции f4 может быть до 3 строк

@print\_result

def f1(arg):

    return list(sorted(list(map(lambda x : x["job-name"], arg))))

@print\_result

def f2(arg):

    return list(filter(lambda x: x.lower().startswith("программист"), arg))

@print\_result

def f3(arg):

    return list(map(lambda x : x +" с опытом Python", arg))

@print\_result

def f4(arg):

    salary = zip(arg,gen\_random(len(arg),100000,200000))

    return [f"{spec}, зарплата {sal} руб." for spec, sal in salary]

if \_\_name\_\_ == '\_\_main\_\_':

    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

**TDD – фреймворк**

**\lab\_03\_04\lab\_05\_TDD.py**

import process\_data as pd

import unittest

test1\_data = [

    {

        "mobile-url": "https://trudvsem.ru/vacancy/card/1027739174033/6bf457e6-51d8-11e6-853e-037acc02728d",

        "description": "<p>Умение общаться по телефону и лично, доброжелательность, ответственность, стрессоустойчивость.</p>",

        "update-date": "2016-10-02 01:33:38 MSK",

        "employment": "Частичная занятость",

        "job-name": "Администратор на телефоне",

        "company": {

            "email": "on.klinik@mail.ru",

            "contact-name": "Светлана",

            "hr-agency": True,

            "phone": "+7(495)6084488",

            "name": "ООО РОЯЛ КЛИНИК"

        },

        "term": "<p>Присутствуют по результатам работы</p>",

        "addresses": {

            "address": {

                "location": "г. Москва, Кузнецкий Мост улица, 1",

                "lat": 55.760808,

                "lng": 37.615713

            }

        },

        "url": "https://trudvsem.ru/vacancy/card/1027739174033/6bf457e6-51d8-11e6-853e-037acc02728d",

        "salary": "от 27000",

        "duty": "<p>Консультирование клиентов по услугам медицинского центра и скидкам. Ориентирование клиента от метро до офиса.</p>",

        "creation-date": "2016-10-02 00:00:00 MSK",

        "requirement": {

            "qualification": "<p>Неполный рабочий день (несколько часов в день) утро/вечер. Стабильные выплаты каждые 2 недели. Работа по договору. Фиксированный оклад 15 000 +бонусы.Дружный коллектив.</p>",

            "education": "Среднее"

        },

        "currency": "«руб.»",

        "schedule": "Неполный рабочий день",

        "category": {

            "industry": "Работы, не требующие квалификации"

        }

    }]

test2\_data = ["Программист","администратор","программист python","ПРОГРАММИСТ","НЕПРОГРАММИСТ","программист крутой"]

class TestMyModule(unittest.TestCase):

    def test\_f1(self):

        self.assertEqual(pd.f1(test1\_data),["Администратор на телефоне"])

    def test\_f2(self):

        self.assertEqual(pd.f2(test2\_data),["Программист","программист python","ПРОГРАММИСТ","программист крутой"])

    def test\_f3(self):

        self.assertEqual(pd.f3(["1","2","3","4"]),["1 с опытом Python","2 с опытом Python","3 с опытом Python","4 с опытом Python"])

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

**BDD – фреймворк**

**\lab\_03\_04\process\_data.feature**

Feature: Process Data

As a user

I want to process data

So that I can get the desired output

Scenario: Use function f1

Given I have a list of dictionaries with resume

When I call function f1 with this list

Then I should get a sorted list of job names

Scenario: Use function f2

Given I have a list of job names

When I call function f2 with this list

Then I should get a list of job names starting with "программист"

Scenario: Use function f3

Given I have a list of job names

When I call function f3 with this list

Then I should get a list of job names with " с опытом Python" appended

**\lab\_03\_04\process\_data\_steps.py**

from behave import given, when, then

from process\_data import f1, f2, f3

@given('I have a list of dictionaries with resume')

def step\_impl(context):

    context.data = [

    {

        "mobile-url": "https://trudvsem.ru/vacancy/card/1027739174033/6bf457e6-51d8-11e6-853e-037acc02728d",

        "description": "<p>Умение общаться по телефону и лично, доброжелательность, ответственность, стрессоустойчивость.</p>",

        "update-date": "2016-10-02 01:33:38 MSK",

        "employment": "Частичная занятость",

        "job-name": "Администратор на телефоне",

        "company": {

            "email": "on.klinik@mail.ru",

            "contact-name": "Светлана",

            "hr-agency": True,

            "phone": "+7(495)6084488",

            "name": "ООО РОЯЛ КЛИНИК"

        },

        "term": "<p>Присутствуют по результатам работы</p>",

        "addresses": {

            "address": {

                "location": "г. Москва, Кузнецкий Мост улица, 1",

                "lat": 55.760808,

                "lng": 37.615713

            }

        },

        "url": "https://trudvsem.ru/vacancy/card/1027739174033/6bf457e6-51d8-11e6-853e-037acc02728d",

        "salary": "от 27000",

        "duty": "<p>Консультирование клиентов по услугам медицинского центра и скидкам. Ориентирование клиента от метро до офиса.</p>",

        "creation-date": "2016-10-02 00:00:00 MSK",

        "requirement": {

            "qualification": "<p>Неполный рабочий день (несколько часов в день) утро/вечер. Стабильные выплаты каждые 2 недели. Работа по договору. Фиксированный оклад 15 000 +бонусы.Дружный коллектив.</p>",

            "education": "Среднее"

        },

        "currency": "«руб.»",

        "schedule": "Неполный рабочий день",

        "category": {

            "industry": "Работы, не требующие квалификации"

        }

    }]

@given('I have a list of job names')

def step\_impl(context):

    context.data = ["Программист","администратор","программист python","ПРОГРАММИСТ","НЕПРОГРАММИСТ","программист крутой"]

@when('I call function f1 with this list')

def step\_impl(context):

    context.result = f1(context.data)

@then('I should get a sorted list of job names')

def step\_impl(context):

    assert context.result == ["Администратор на телефоне"]

@when('I call function f2 with this list')

def step\_impl(context):

    context.result = f2(context.data)

@then('I should get a list of job names starting with "программист"')

def step\_impl(context):

    assert context.result == ["Программист","программист python","ПРОГРАММИСТ","программист крутой"]

@when('I call function f3 with this list')

def step\_impl(context):

    context.result = f3(context.data)

@then('I should get a list of job names with " с опытом Python" appended')

def step\_impl(context):

    assert context.result == ["Программист с опытом Python","администратор с опытом Python","программист python с опытом Python","ПРОГРАММИСТ с опытом Python","НЕПРОГРАММИСТ с опытом Python","программист крутой с опытом Python"]

**Анализ результатов**



