Московский Государственный Технический Университет им. Н.Э. Баумана

Отчет по лабораторной работе № 5 по курсу Базовые компоненты интернет-технологий

" Модульное тестирование в Python."

Исполнитель:

Викулин Иван Сергеевич, РТ5-31Б

Проверил:

Гапанюк Юрий Евгеньевич

Москва, 2022

**Задание**

Цель лабораторной работы: изучение возможностей модульного тестирования в языке Python.

Задание:

Выбрать любой фрагмент кода из предыдущих лабораторных работ (выбранная работа: 3-4 лабораторная).

Модифицировать код таким образом, чтобы он был пригоден для модульного тестирования.

Разработать модульные тесты с помощью TDD и BDD фреймворков.

**Код**

**cm\_timer.py:**

from contextlib import contextmanager

from time import time, sleep

class cm\_timer\_1:

    def \_\_init\_\_(self):

        self.start\_time = time()

    def \_\_enter\_\_(self):

        return self

    def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

        if exc\_type is not None:

            print(exc\_type, exc\_val, exc\_tb)

        self.end\_time = time()

        print(f'Время работы блока (class) {self.end\_time - self.start\_time:.5f} сек.')

@contextmanager

def cm\_timer():

    start\_time = time()

    yield 1

    print(f'Время работы блока (function) {time() - start\_time:.5f} сек.')

# with cm\_timer():

#     sleep(1.5)

# with cm\_timer\_1():

#     sleep(1.5)

**field.py:**

goods = [

    {'title': 'Ковер', 'price': 2000, 'color': 'green'},

    {'title': 'Диван для отдыха', 'color': 'black'}

]

def field(items, \*args):

    assert len(args) > 0

    result = []

    for item in items:

        dict\_result = {}

        if len(args) == 1 and item[args[0]] is not None:

            result.append(item[args[0]])

        else:

            for key in args:

                if key in item and item[key] is not None:

                    dict\_result[key] = item[key]

        if len(dict\_result) > 0:

            result.append(dict\_result)

    return result

# print(field(goods, 'title'))

# print(field(goods, 'title', 'price'))

# print(field(goods, 'title', 'price', 'color'))

# print(field(goods, 'title', 'color'))

**gen\_random.py:**

import random

def gen\_random(num\_count, begin, end):

    result = [random.randint(begin, end) for i in range(num\_count)]

    return result

# print(gen\_random(5, 1, 3))

# print(gen\_random(5, 5, 10))

**print\_result.py:**

def print\_result(func):

    def wrapper(\*args, \*\*kwargs):

        print(func.\_\_name\_\_)

        result = func(\*args, \*\*kwargs)

        if (isinstance(result, list)):

            print(\*result, sep = '\n')

        elif (isinstance(result, dict)):

            for key, value in result.items():

                print(key, value, sep = ' = ')

        else:

            print(result)

        return result

    return wrapper

@print\_result

def test\_1():

    return 1

@print\_result

def test\_2():

    return 'iu5'

@print\_result

def test\_3():

    return {'a': 1, 'b': 2}

@print\_result

def test\_4():

    return [1, 2]

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

    print('!!!!!!!!')

    test\_1()

    test\_2()

    test\_3()

    test\_4()

    print('!!!!!!!!')

**sort.py:**

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]

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

    result = sorted(data, key=abs, reverse=True)

    print(result)

    result\_with\_lambda = sorted(data, key = lambda x: -abs(x))

    print(result\_with\_lambda)

**unique.py:**

#from gen\_random import gen\_random

class Unique(object):

    def \_\_init\_\_(self, items, \*\*kwargs):

        self.items = list(items)

        self.ignore\_case = kwargs.get('ignore\_case', False)

        self.unique\_items = set()

    def \_\_iter\_\_(self):

        return self

    def \_\_next\_\_(self):

        if len(self.items) == 0:

            raise StopIteration

        item = self.items.pop(0)

        if self.ignore\_case:

            item = item.lower()

        if item not in self.unique\_items:

            self.unique\_items.add(item)

            return item

        else:

            return self.\_\_next\_\_()

# data = [1,1,1,2,2,3,4,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]

# print(\*Unique(data))

# data = ['a', 'A', 'B', 'a', 'b']

# print(\*Unique(data, ignore\_case=True))

# print(\*Unique(data))

**process\_data.py:**

import json

import sys

from cm\_timer import cm\_timer\_1

from gen\_random import gen\_random

from print\_result import print\_result

from field import field

from unique import Unique

path = 'lab3\_4/lab\_python\_fp/data\_light.json'

with open(path, encoding='UTF8') as f:

    data = json.load(f)

@print\_result

def f1(arg):

    return sorted(Unique(field(arg, 'job-name'), ignore\_case=True))

    # return (Unique(sorted(field(arg, 'job-name'), key=len), ignore\_case=True))

@print\_result

def f2(arg):

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

@print\_result

def f3(arg):

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

@print\_result

def f4(arg):

    return list(map(lambda x: x + ', зарплата ' + str(gen\_random(1, 100000, 200000)) + ' руб.', arg))

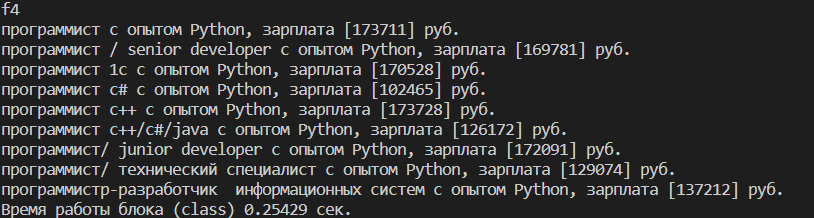
if \_\_name\_\_ == '\_\_main\_\_':

    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

**Результаты выполнения программы**

*Основной результат:*



*Промежуточные функции:*

