**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 Лабораторной работе №3-4

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б: |  | преподаватель каф. ИУ5 |
| Лапотко Ирина Евгеньевна |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2022 г.

**Постановка задачи:**

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab\_python\_fp. Решение каждой задачи должно раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

**Текст программы:**

**cm-timer.py**

from contextlib import contextmanager

import time

class cm\_timer\_1:

    def \_\_init\_\_(self):

        pass

    def \_\_enter\_\_(self):

        self.time = time.time()

    def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):

        if exp\_type is not None:

            print(exp\_type, exp\_value, traceback)

        else:

            print(time.time() - self.time)

@contextmanager

def cm\_timer\_2():

    time\_=time.time()

    yield 7

    print(time.time() - time\_)

with cm\_timer\_1():

    time.sleep(5.5)

with cm\_timer\_1():

    time.sleep(4.5)

**field.py**

def field(items, \*args):

    assert len(args)>0

    if len(args) == 1:

        for i in items:

            a = i.get(args[0])

            if a != None:

                yield a

    else:

        for i in items:

            new\_dict = {}

            for j in args:

                a = i.get(j)

                if a != None:

                    new\_dict[j] = a

            yield new\_dict

    return 0

def main():

    goods = [

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

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

    a = field(goods, 'title','price')

    b = field(goods, 'title')

    for i in a:

        print(i, end = ' ')

    print('\n')

    for i in b:

        print(i, end = ' ')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**print\_result.py**

def print\_result(func):

    def wrapper(\*args):

        ret = func(\*args)

        name = func.\_\_name\_\_

        print(name)

        if type(ret) == list:

            for i in ret:

                print(i)

        elif type(ret) == dict:

            for i in ret.keys():

                    print (i, ' = ', ret[i])

        else:

            print (ret)

        return ret

    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()

**get\_random.py**

from random import randint

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

    for i in range(num\_count):

        yield randint(begin, end)

def main():

    a = get\_random(5, 1 , 3)

    for i in a:

        print(i, end = ' ')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**sort.py**

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

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

    print(data)

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

    print(result)

    f = lambda x: sorted(x, key = abs, reverse = True)

    result\_with\_lambda = f(data)

    print(result\_with\_lambda)

**Unique.py**

class Unique(object):

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

        self.data = items

        self.index = 0

        self.res\_data = set()

        if kwargs.get('ignore\_case') == None: #ignore\_case ключ

            self.ignore\_case = False

        else:

            self.ignore\_case = kwargs['ignore\_case']

    def \_\_next\_\_(self):

        while True:

            if self.index >= len(self.data):

                raise StopIteration

            else:

                current = self.data[self.index]

                self.index = self.index + 1

                if self.ignore\_case == False:

                    if current not in self.res\_data:

                        self.res\_data.add(current)

                        return current

                else:

                    low\_data = [i.lower() for i in self.res\_data]

                    if current.lower() not in low\_data:

                        self.res\_data.add(current)

                        return current

    def \_\_iter\_\_(self):

        return self

data = [1,4,1,8]

for i in Unique(data):

    print(i, end = ' ')

**Process\_data.py**

import json

import sys

# Сделаем другие необходимые импорты

from cm\_timer import cm\_timer\_1

from field import field

from get\_random import get\_random

from print\_result import print\_result

import sort

from unique import Unique

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

path = 'data\_light.json'

with open(path,'r', encoding = 'utf8') 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 sorted(Unique([data[i]['job-name'] for i in range(len(data))], ignore\_case = True))

@print\_result

def f2(arg):

    return list(filter(lambda x: x[:11].lower()=='программист',arg))

@print\_result

def f3(arg):

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

@print\_result

def f4(arg):

    pay = get\_random(len(arg), 100000, 200000)

    res = list(zip(arg, pay))

    return [res[i][0] + ',зарплата' + str(res[i][1]) + 'руб. ' for i in range(len(arg))]

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

    with cm\_timer\_1():

        f4(f3(f2(f1(data))))