**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Лабораторная работа №3-4

Выполнил:

Студент ИУ5-34Б

Флоринский В. А.

Подпись и дата:

Проверил:

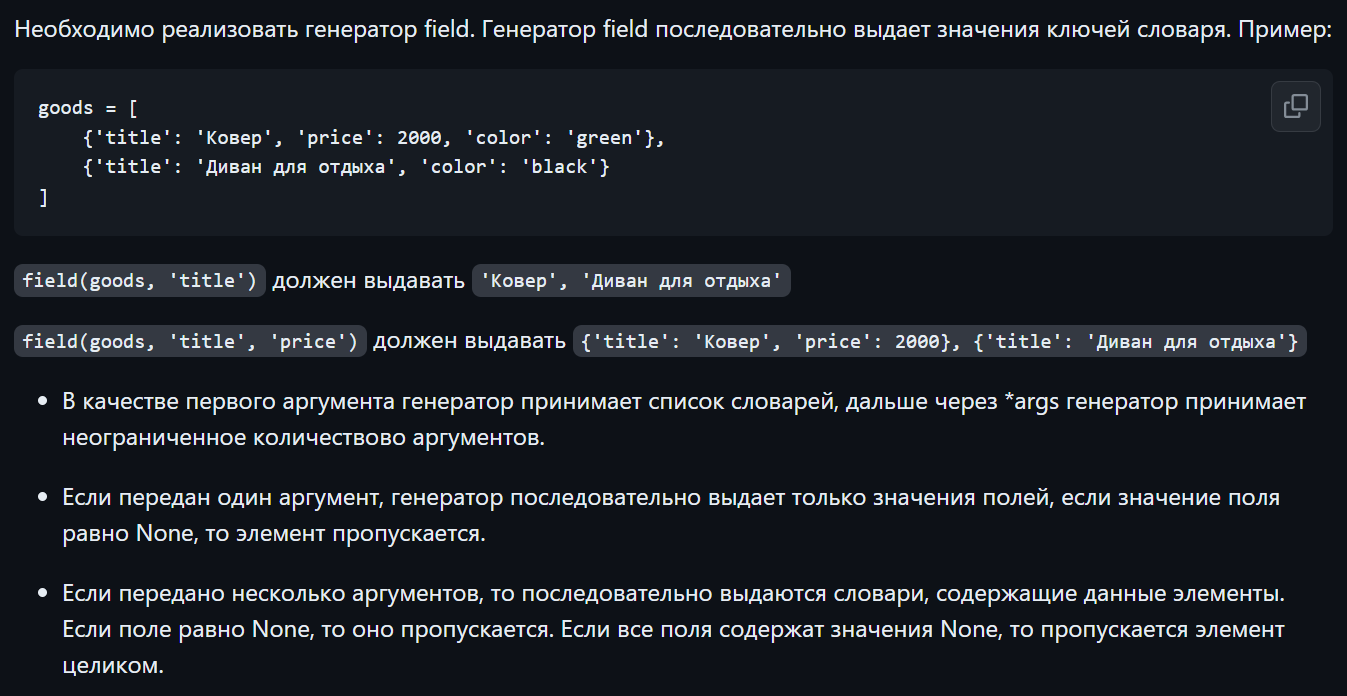
Преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2023 г.

**Задание 1.**

****

**Текст программы.**

**field.py:**

def field(items, \*spec, key=0):

    assert len(spec) > 0, "Параметры обязательно указывать"

    if len(spec) == 1:

        ans = []

        for i in range(len(items)-1):

            if items[i].get(spec[0]) != None : ans.append(items[i].get(spec[0]))

        if items[i+1].get(spec[0]) != None : ans.append(items[i+1].get(spec[0]))

        return ans

    else:

        for it in items:

            tdict = {}

            for sp in spec:

                if it.get(sp) != None : tdict[sp] = it.get(sp)

            if len(tdict.keys()) != 0 : return(tdict)

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

    cars = [

        {'name' : 'bmw', 'price' : 10000, 'max\_speed' : 300, 'hpower' : 650},

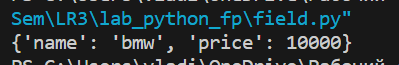
        {'name' : 'audi', 'price' : 9000, 'max\_speed' : 250, 'hpower' : 400, 'color' : 'red'},

        {'name' : 'mercedes', 'price' : 15000, 'max\_speed' : 270, 'hpower' : 4000, 'color' : 'green'}

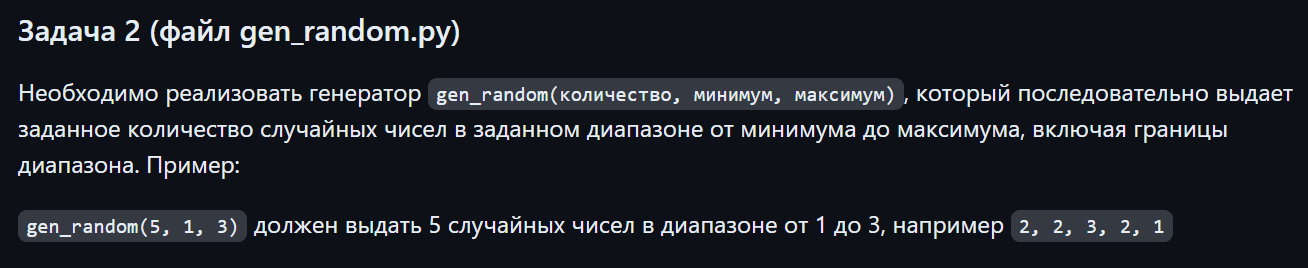
    ]

    print(field(cars, 'name', 'price'))

**Примеры работы.**



**Задание 2.**

****

**Текст программы.**

**gen\_random.py:**

from random import randint

def gen\_random(count\_, min\_, max\_):

    arr = [randint(min\_, max\_) for i in range(count\_)]

    return arr

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

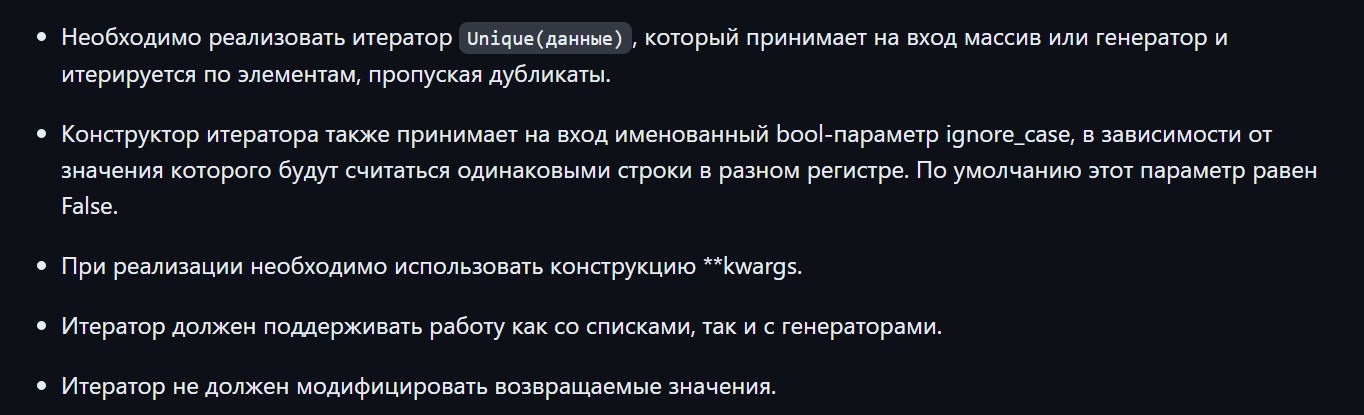
    arr = gen\_random(5, 1, 3)

    print(arr)

**Примеры работы.**

****

**Задание 3.**

****

**Текст программы.**

**Unique.py:**

class Unique(object):

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

        self.items = items

        self.curr = 0

        if len(list(kwargs.keys())) == 0 or list(kwargs.values()) == [False]:

            self.ign = False #a != A

            self.set\_arr = list(set(self.items))

        elif list(kwargs.values()) == [True]:

            self.ign = True #a == A

            self.set\_arr = []

            for i in self.items:

                if (type(i) is str) and (i.upper() not in self.set\_arr and i.lower() not in self.set\_arr):

                    self.set\_arr.append(i)

                elif i not in self.set\_arr and (type(i) is not str): self.set\_arr.append(i)

    def \_\_next\_\_(self):

        if self.curr < len(self.set\_arr):

            res = self.set\_arr[self.curr]

            self.curr += 1

            return res

        raise StopIteration

    def \_\_iter\_\_(self):

        return self

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

    arr = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    #arr = [1,2,2,3,3,1]

    it = Unique(arr, ignore\_case = True)

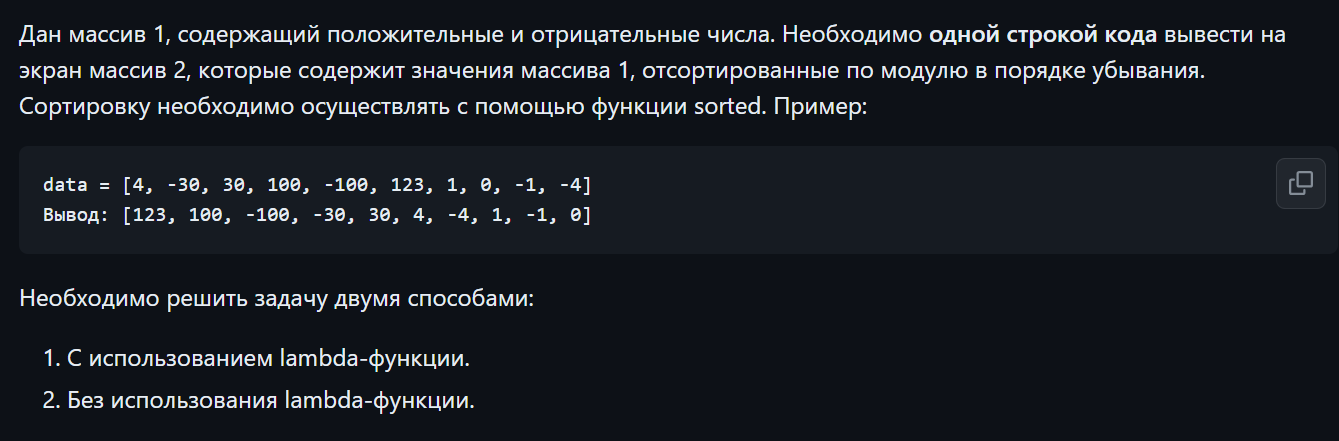
    for el in it:

        print(el)

**Примеры работы.**

****

**Задание 4.**

****

**Текст программы.**

**sort.py:**

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

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

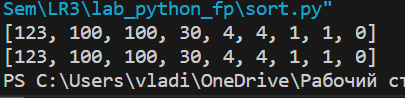
    result\_with\_lambda = sorted(list(map(lambda i: abs(i), data)), reverse=True)

    print(result\_with\_lambda)

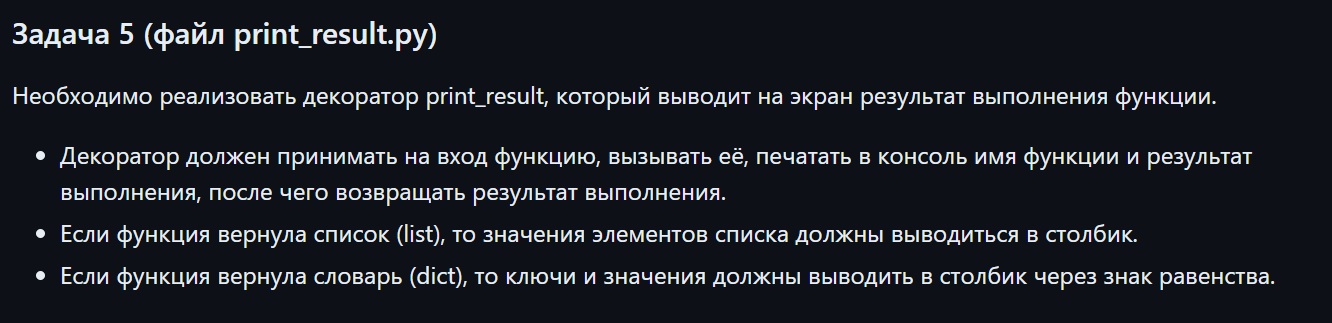
    result = sorted(list(map(abs, data)), reverse=True)

    print(result)

**Примеры работы.**

****

**Задание 5.**

****

**Текст программы.**

**print\_result.py:**

def print\_result(func):

    def wrapper(\*args, \*\*kwargs):

        print(func.\_\_name\_\_)

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

        if type(temp) is list:

            for el in temp: print(el)

        elif type(temp) is dict:

            for key,value in temp.items(): print(f"{key} = {value}")

        else: print(temp)

        return temp

    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('!!!!!!!!')

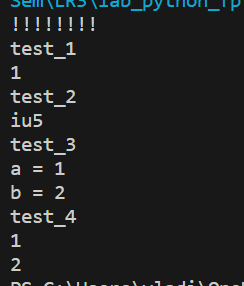
    test\_1()

    test\_2()

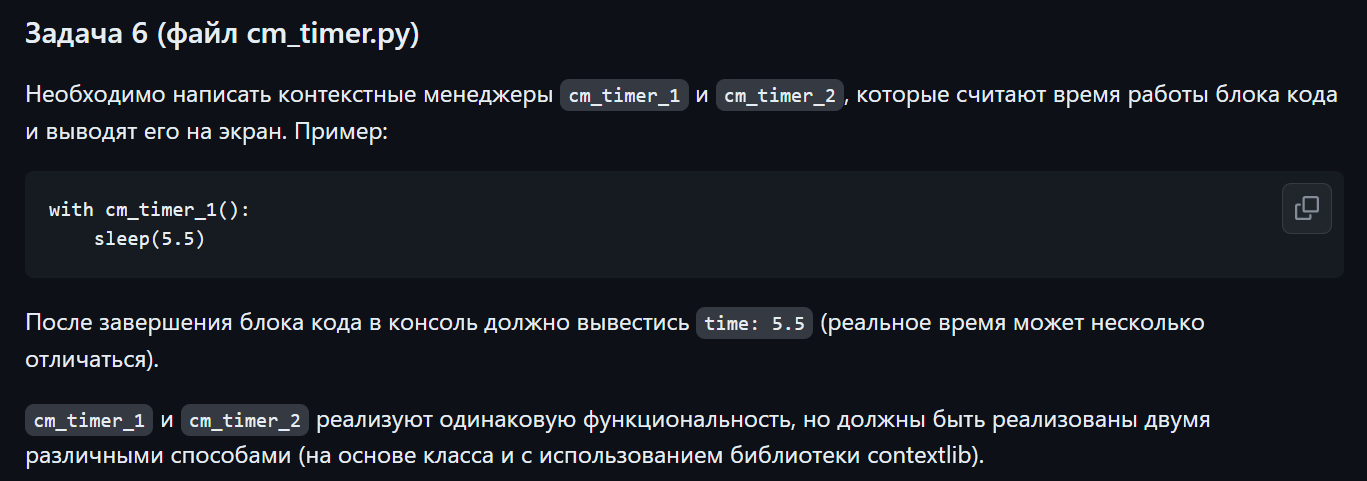
    test\_3()

    test\_4()

**Результат работы.**

****

**Задание 6.**

****

**Текст программы.**

**cm\_timer.py:**

from contextlib import contextmanager

import time

class Cm\_timer\_1:

    def \_\_enter\_\_(self):

        self.start\_time = time.time()

        return self

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

        delta\_time = time.time() - self.start\_time

        print(f"time: {round(delta\_time, 3)}")

@contextmanager

def Cm\_timer\_2():

    start\_time = time.time()

    try:

        yield

    finally:

        delta\_time = time.time() - start\_time

        print(f"time: {round(delta\_time, 3)}")

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

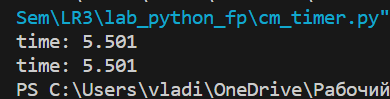
    with Cm\_timer\_1():

        time.sleep(5.5)

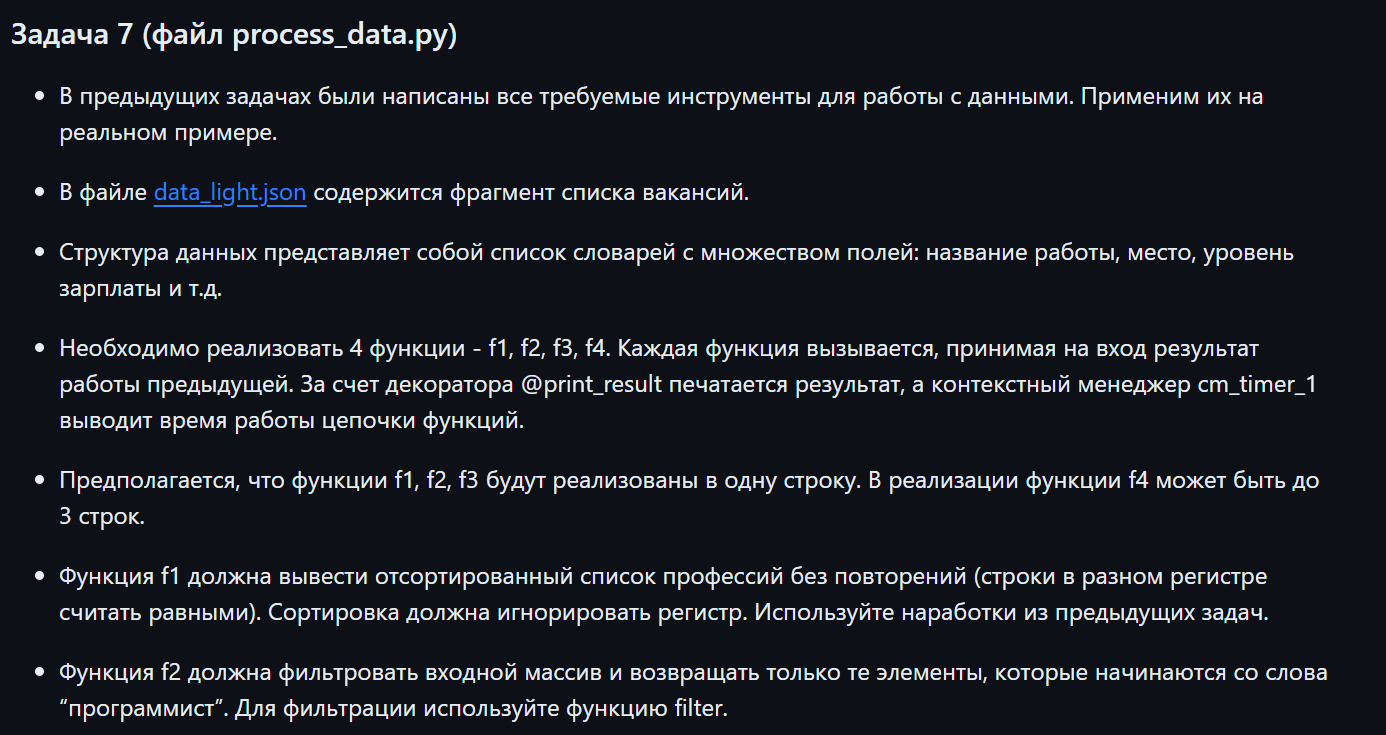
    with Cm\_timer\_2():

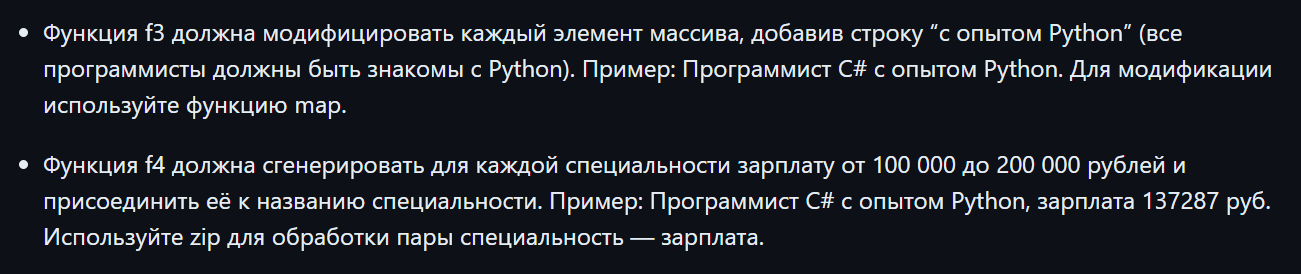
        time.sleep(5.5)

**Результат работы.**

****

**Задание 7.**

****

****

**Текст программы.**

**process\_data.py:**

import json

from random import randint

from unique import Unique

from print\_result import print\_result

from field import field

from gen\_random import gen\_random

from cm\_timer import Cm\_timer\_1

path = "C:\\Users\\vladi\\OneDrive\\Рабочий стол\\LR3thSem\\LR3\\lab\_python\_fp\\data\_light.json"

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

    data = json.load(f)

@print\_result

def f1(arg):

    return sorted([el for el in Unique(field(arg, 'job-name'), ignore\_case = True)])

@print\_result

def f2(arg):

    return list(filter(lambda x: x.split()[0] == "программист", arg))

@print\_result

def f3(arg):

    return list(map(lambda x: f"{x} с опытом Python", arg))

@print\_result

def f4(arg):

    return  [f"{man}, зарплата {sal} руб." for man, sal in zip(arg, gen\_random(len(arg), 100000, 200000))]

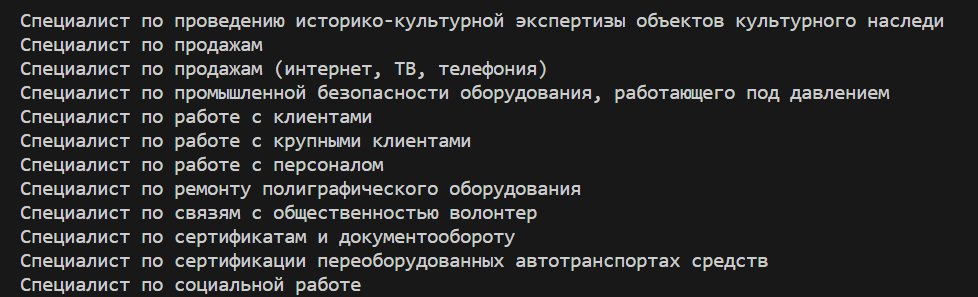
if \_\_name\_\_ == "\_\_main\_\_":

    with Cm\_timer\_1():

        f4(f3(f2(f1(data))))

**Результат работы.**

**Результат функции f1():**



Выводит все профессии без повторений

Результаты остальных функций:

