**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №3

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-51Б |  | преподаватель каф.ИУ5 |
| Грызин Алексей |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

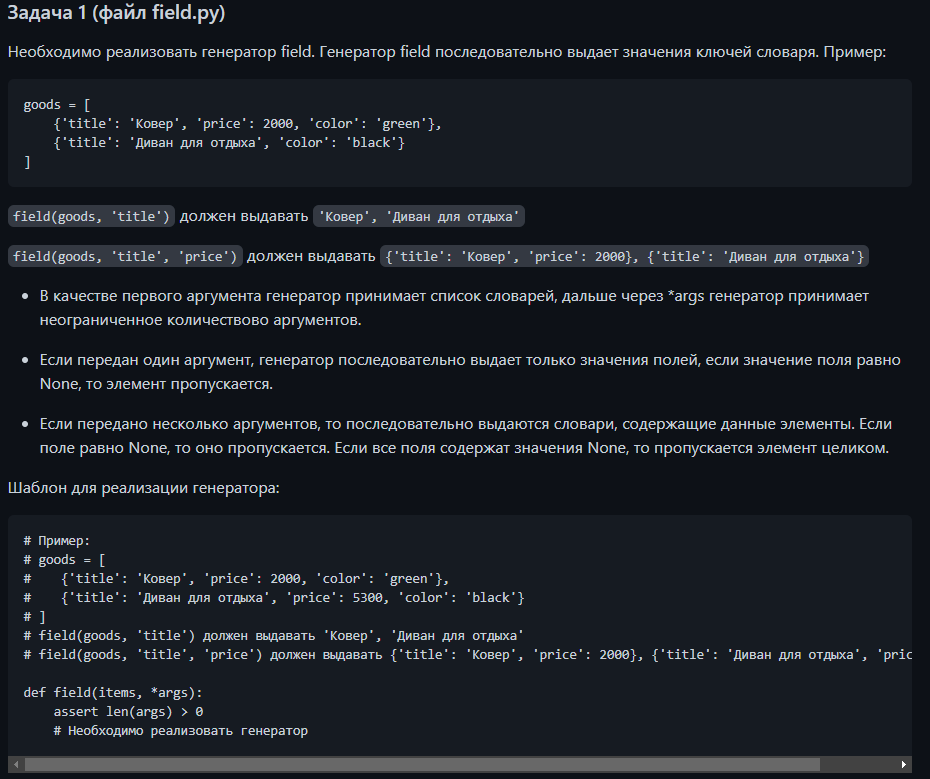
# Описание задания

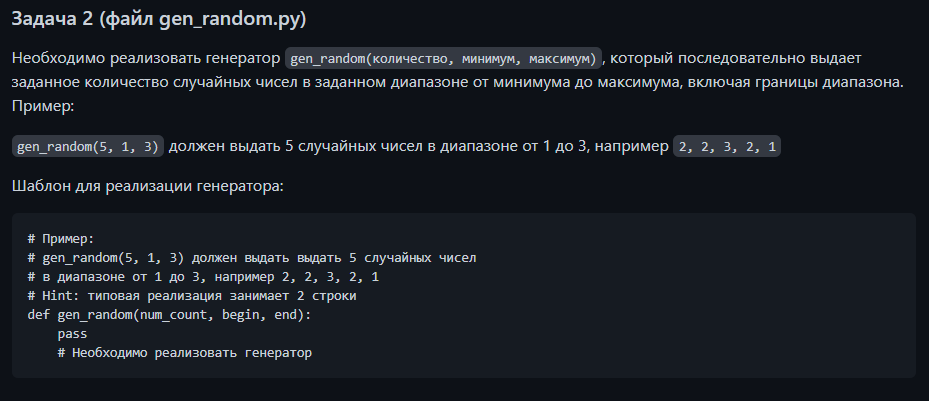
**Цель лабораторной работы**: изучение возможностей функционального программирования в языке Python.

Задание лабораторной работы состоит из решения нескольких задач.

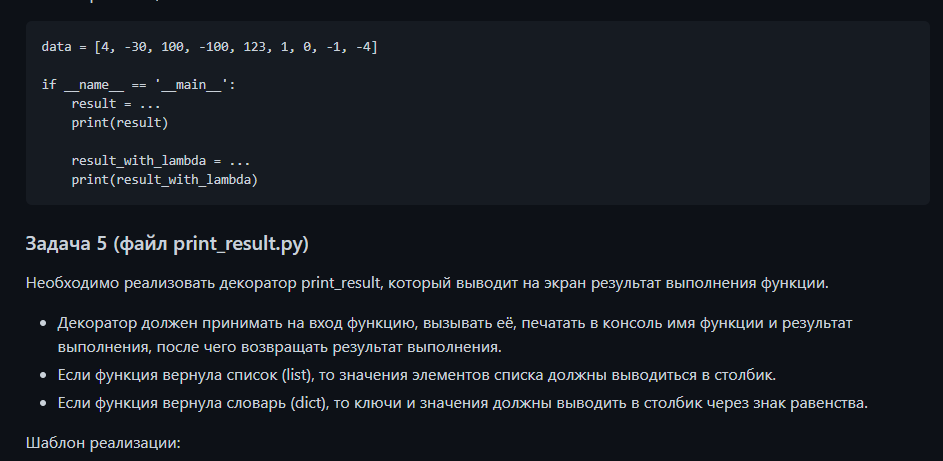
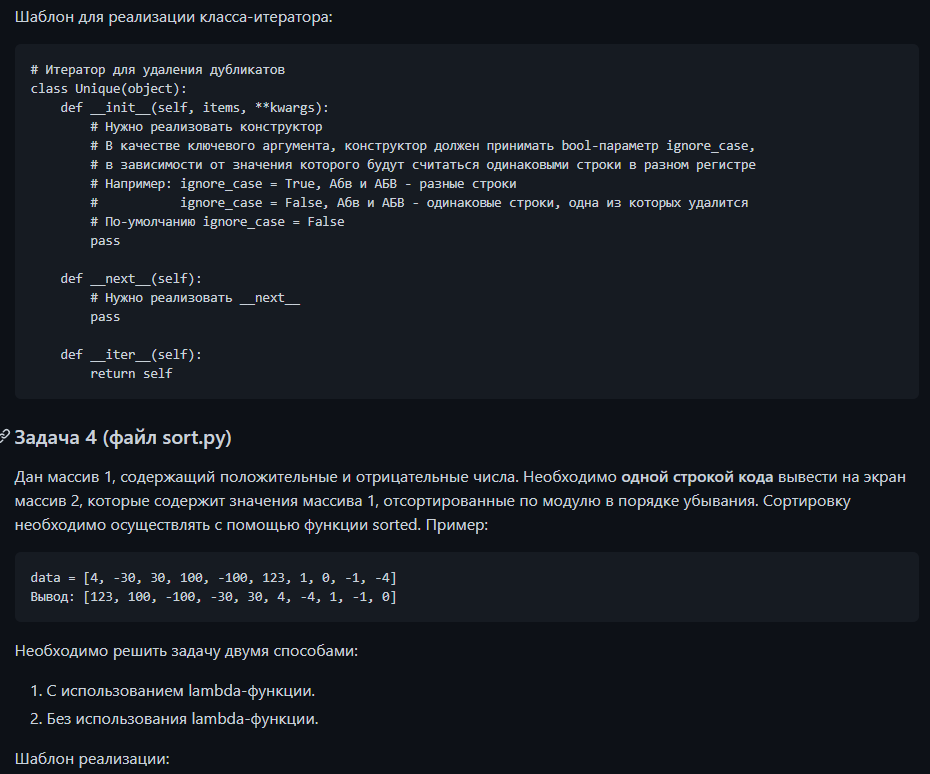
Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab\_python\_fp. Решение каждой задачи должно раполагаться в отдельном файле.

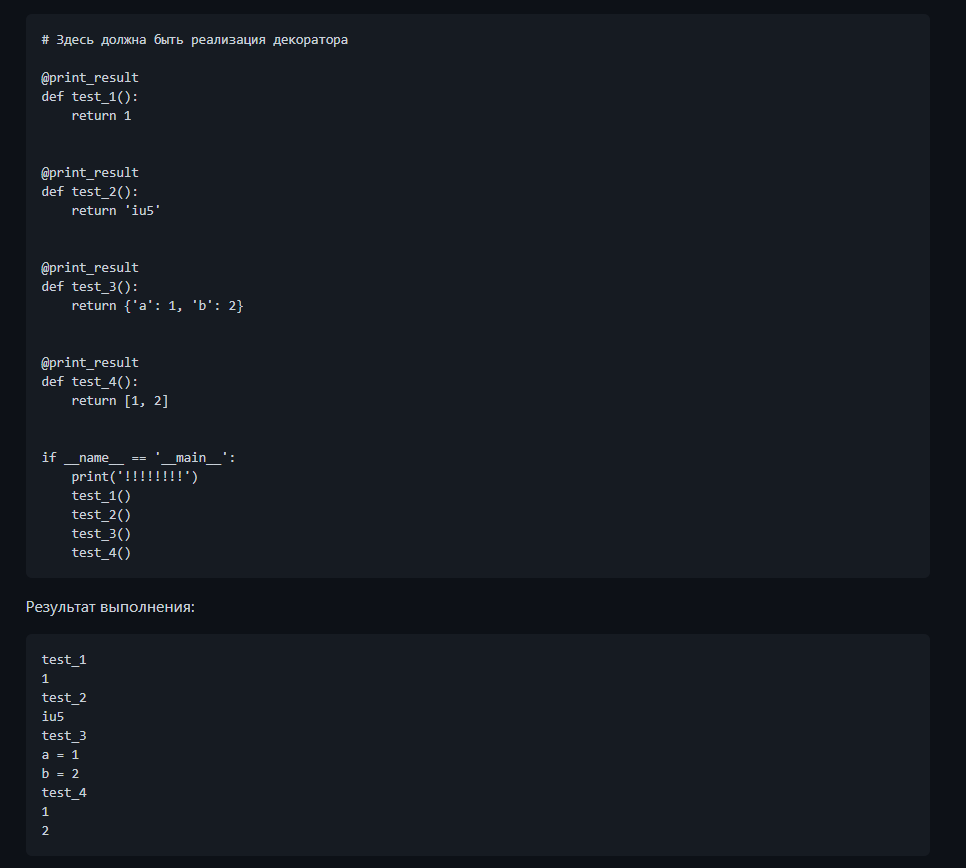
При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

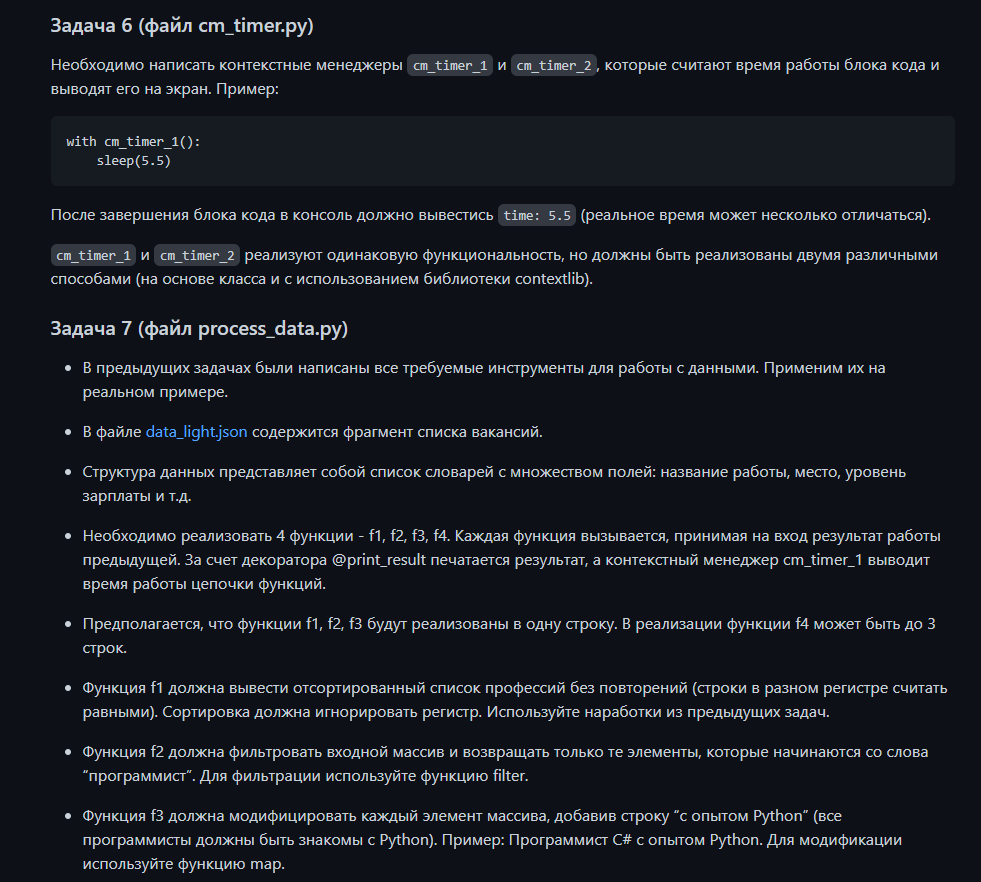


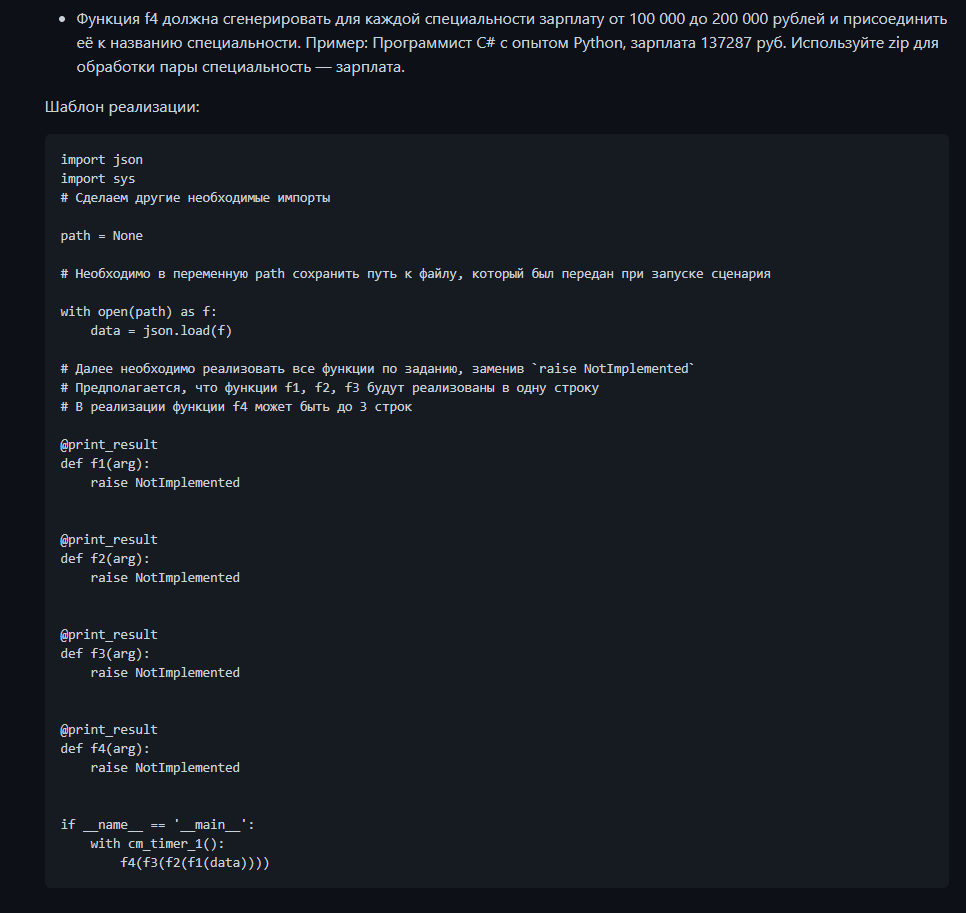












# Тексты программ

***Задача 1:*** field.py

from typing import Dict, List

def **field**(dicts: List[Dict], \*args: List[str]) -> List:

    assert **len**(args) > 0 and **len**(dicts) > 0

    return\_value = []

    for dict in dicts:

        buf\_dict = {}

        for key in dict.**keys**():

            if **len**(args) == 1:

                if key in args:

                    return\_value.**append**(dict[key])

            else:

                if key in args:

                    buf\_dict[key] = dict[key]

        if **len**(args) > 1: return\_value.**append**(buf\_dict)

    return return\_value

goods = [

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

    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black', 'material': 'Кожа'}

]

data\_int = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

data\_str = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

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

def **test\_field**():

**print**('Task 1-1: ', str(**field**(goods, 'title'))[1:-1])

**print**('Task 1-2: ', str(**field**(goods, 'title', 'price'))[1:-1])

**print**('Task 1-3: ', str(**field**(goods, 'title', 'price', 'material'))[1:-1])

**print**('-------------------------------------------------------')

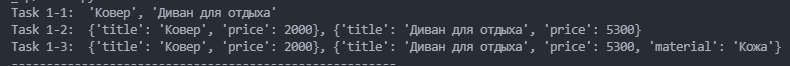
def **main**():

**test\_field**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***



***Задача 2:*** gen\_random.py

from typing import List

from random import **randint**

def **gen\_random**(num\_count, begin, end) -> List:

    assert num\_count >= 0

    return [**randint**(begin, end) for i in range(num\_count)]

def **test\_gen\_random**():

**print**('Task 2-1: ', str(**gen\_random**(5, 1, 3))[1:-1])

**print**('Task 2-2: ', str(**gen\_random**(3, 1, 100))[1:-1])

**print**('Task 2-3: ', str(**gen\_random**(5, -100, 100))[1:-1])

**print**('-------------------------------------------------------')

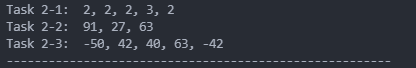
def **main**():

**test\_gen\_random**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***



***Задача 3:*** unique.py

# Итератор для удаления дубликатов

from gen\_random import **gen\_random**

class Unique(object):

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

        self.used\_elements = set()

        self.data = list(items)

        self.index = 0

        if 'ignore\_case' in kwargs.**keys**():

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

        else:

            self.ignore\_case = False

    def **\_\_next\_\_**(self):

        while True:

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

                raise StopIteration

            current = self.data[self.index]

            self.index += 1

            if ((self.ignore\_case or not **isinstance**(current, str)) and current not in self.used\_elements):

                self.used\_elements.**add**(current)

                return current

            elif (not self.ignore\_case and **isinstance**(current, str) and current.**upper**() not in self.used\_elements

                    and current.**lower**() not in self.used\_elements):

                self.used\_elements.**add**(current.**upper**())

                self.used\_elements.**add**(current.**lower**())

                return current

    def **\_\_iter\_\_**(self):

        return self

data\_int = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

data\_str = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

def **test\_unique**():

**print**('Task 3-1: ', str(list(Unique(data\_int)))[1:-1])

**print**('Task 3-2 ignoring case: ', str(list(Unique(data\_str, ignore\_case = True)))[1:-1])

**print**('Task 3-2 not ignoring case: ', str(list(Unique(data\_str, ignore\_case = False)))[1:-1])

**print**('Task 3-3: ', str(list(Unique(**gen\_random**(100, 1, 5))))[1:-1])

**print**('-------------------------------------------------------')

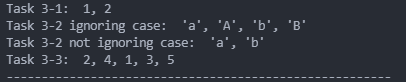
def **main**():

**test\_unique**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***



***Задача 4:*** sort.py

def **sort**(data):

    return **sorted**(data, key=**abs**, reverse=True)

def **sort\_lambda**(data):

    return **sorted**(data, key=lambda value: value if value > 0 else -value, reverse=True)

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

def **test\_sort**():

**print**('Task 4 without lambda: ', str(**sort**(data\_sort))[1:-1])

**print**('Task 4 with lambda: ', str(**sort\_lambda**(data\_sort))[1:-1])

**print**('-------------------------------------------------------')

def **main**():

**test\_sort**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***



***Задача 5:*** print\_result.py

def **print\_result**(func):

    def **decorated\_func**(\*args):

**print**(func.\_\_name\_\_)

        return\_value = func(\*args)

        if **isinstance**(return\_value, list):

            for value in return\_value:

**print**(str(value))

        elif **isinstance**(return\_value, dict):

            for key in return\_value.**keys**():

**print**(str(key) + ' = ' + str(return\_value[key]))

        else:

**print**(return\_value)

        return return\_value

    return **decorated\_func**

def **test\_print\_result**():

**@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]

**print**('Task 5:\n')

**test\_1**()

**test\_2**()

**test\_3**()

**test\_4**()

**print**('-------------------------------------------------------')

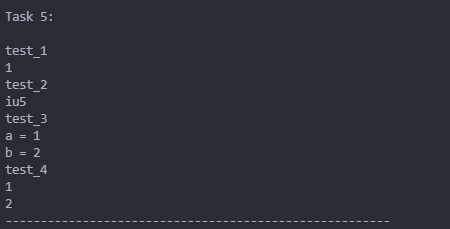
def **main**():

**test\_print\_result**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***



***Задача 6:*** cm\_timer.py

import time

from contextlib import **contextmanager**

class cm\_timer\_1:

    def **\_\_enter\_\_**(self):

        self.start\_time = time.**time**()

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

**print**(cm\_timer\_1.\_\_name\_\_, time.**time**() - self.start\_time)

**@contextmanager**

def **cm\_timer\_2**():

    start\_time = time.**time**()

    yield

**print**(**cm\_timer\_2**.\_\_name\_\_, time.**time**() - start\_time)

def **test\_timer**():

**print**('Task 6:\n')

    with cm\_timer\_1():

        time.**sleep**(5.5)

    with **cm\_timer\_2**():

        time.**sleep**(5.5)

**print**('-------------------------------------------------------')

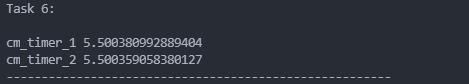
def **main**():

**test\_timer**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***



***Задача 7:*** process\_data.py

import json

from cm\_timer import cm\_timer\_1

from field import **field**

from gen\_random import **gen\_random**

from print\_result import **print\_result**

from unique import Unique

path = 'lab3/json/data\_light.json'

def **process\_data**():

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

        data = json.**load**(f)

**@print\_result**

    def **f1**(value):

        return **sorted**(Unique(**field**(value, 'job-name')))

**@print\_result**

    def **f2**(value):

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

**@print\_result**

    def **f3**(value):

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

**@print\_result**

    def **f4**(value):

        salary = list(**gen\_random**(**len**(value), 100000, 200000))

        return list(map(lambda x: x[0] + ', зарплата ' + str(x[1]) + ' руб', list(zip(value, salary))))

    with cm\_timer\_1():

**f4**(**f3**(**f2**(**f1**(data))))

def **test\_process\_data**():

**print**('Task 7:\n')

**process\_data**()

def **main**():

**test\_process\_data**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

***Результат***

