**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

 **(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Отчёт по лабораторной работе №3:

«Функциональные возможности языка Python»

Выполнилa:

Студентка группы ИУ5-55Б

Е. И. Мащенко

***2020***

***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.

**Задача 1 (файл field.py)**

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

* В качестве первого аргумента генератор принимает список словарей, дальше через \*args генератор принимает неограниченное количествово аргументов.
* Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
* Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

**Задача 2 (файл gen\_random.py)**

Необходимо реализовать генератор gen\_random(количество, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона.

**Задача 3 (файл unique.py)**

* Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
* Конструктор итератора также принимает на вход именованный bool-параметр ignore\_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False.
* При реализации необходимо использовать конструкцию \*\*kwargs.
* Итератор должен поддерживать работу как со списками, так и с генераторами.
* Итератор не должен модифицировать возвращаемые значения.

**Задача 4 (файл sort.py)**

Дан массив 1, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив 2, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted.

Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

**Задача 5 (файл print\_result.py)**

* Необходимо реализовать декоратор print\_result, который выводит на экран результат выполнения функции.
* Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
* Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
* Если функция вернула словарь (dict), то ключи и значения должны выводить в столбик через знак равенства.

**Задача 6 (файл cm\_timer.py)**

Необходимо написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, которые считают время работы блока кода и выводят его на экран.

cm\_timer\_1 и cm\_timer\_2 реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами.

**Задача 7 (файл process\_data.py)**

* В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
* В файле [data\_light.json](https://github.com/iu5team/iu5web-fall-2020/tree/master/labs/lab3/data/data_light.json) содержится фрагмент списка вакансий.
* Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
* Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print\_result печатается результат, а контекстный менеджер cm\_timer\_1 выводит время работы цепочки функций.
* Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
* Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
* Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
* Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python). Пример: Программист C# с опытом Python. Для модификации используйте функцию map.
* Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: Программист C# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

***Текст программы***

field.py

goods = [

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

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

    {'title': 'Стол', 'price': None, 'color': 'white'},

]

def field(items, \*args):

    values=[]

    assert len(args) > 0

    if len(args)==1:

        for item in items:

            val=item.get(args[0])

            if (val!=None):

                values.append(val)

        return(values)

    else:

        for item in items:

            dic={}

            for arg in args:

                val=item.get(arg)

                if (val!=None):

                    dic[arg]=val

            if (len(dic)!=0):

                values.append(dic)

        return(values)

def main():

    print('Пример 1:')

    print(field(goods, 'title'))

    print('\n','Пример 2:')

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

if \_\_name\_\_ == "\_\_main\_\_":

    main()

gen\_random.py

import random

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

    for i in range(num\_count):

        yield random.randint(begin, end)

def main():

    gen = gen\_random(5, 1, 3)

    for i in gen:

        print(i, end=' ')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

unique.py

from lab\_python\_fp.gen\_random import gen\_random

class Unique(object):

    """Итератор, оставляющий только уникальные значения."""

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

        self.used\_elements = set()

        self.items = items

        self.index = 0

        if len(kwargs)!=0:

            self.ignore\_case=kwargs

        else:

            self.ignore\_case=False

    def \_\_iter\_\_(self):

        return self

    def \_\_next\_\_(self):

        while True:

                for item in self.items:

                    current = item

                    self.index = self.index + 1

                    if (current not in self.used\_elements)\

                        and not(self.ignore\_case and current.swapcase() in self.used\_elements):

                        self.used\_elements.add(current)

                        return current

                else:

                    raise StopIteration

def main():

    data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

    print(data1)

    itr1 = Unique(data1)

    for i1 in itr1:

        print(i1, end=' ')

    print('\n', end='')

    data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    print(data2)

    itr2 = Unique(data2)

    for i2 in itr2:

        print(i2, end=' ')

    print('\n', end='')

    print(data2)

    itr3 = Unique(data2, ignor\_case=True)

    for i3 in itr3:

        print(i3, end=' ')

    print('\n', end='')

    data3 = gen\_random(5, 1, 3)

    itr4 = Unique(data3)

    for i4 in itr4:

        print(i4, end=' ')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

sort.py

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

def absolute(x):

    return abs(x)

if \_\_name\_\_ == '\_\_main\_\_':

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

    print(result)

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

    print(result\_with\_lambda)

print\_result.py

#декоратор

def print\_result(func\_to\_decorate):

     def decorated\_func(\*args):

        print(func\_to\_decorate.\_\_name\_\_)

        result = func\_to\_decorate(\*args)

        if type(result) is list:

            for i in result:

                print(i)

        elif type(result) is dict:

            for i in result:

                print(i,' = ', result.get(i))

        else:

            print(result)

        return result

     return decorated\_func

@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()

cm\_timer.py

import time

from contextlib import contextmanager

class cm\_timer\_1:

    def \_\_init\_\_(self):

        self.start\_time = time.time()

    def \_\_enter\_\_(self):

        pass

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

        if exc\_type is not None:

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

        else:

            print('time: ', time.time() - self.start\_time)

@contextmanager

def cm\_timer\_2():

    start\_time = time.time()

    yield 333

    print('time: ', time.time() - start\_time)

def main():

    with cm\_timer\_1():

        time.sleep(5.5)

    with cm\_timer\_2():

        time.sleep(2.5)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

process\_data.py

from lab\_python\_fp.cm\_timer import cm\_timer\_1

from lab\_python\_fp.print\_result import print\_result

from lab\_python\_fp.unique import Unique

from lab\_python\_fp.field import field

from lab\_python\_fp.gen\_random import gen\_random

import json

import sys

import os

THIS\_FOLDER = os.path.dirname(os.path.abspath(\_\_file\_\_))

my\_file = os.path.join(THIS\_FOLDER, 'data\_light.json')

with open(my\_file, encoding='utf-8') as f:

    data = json.load(f)

@print\_result

def f1(arg):

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

@print\_result

def f2(arg):

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

@print\_result

def f3(arg):

    print(arg)

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

@print\_result

def f4(arg):

    salaries = gen\_random(len(arg), 100000, 200000)

    res = list(zip(arg, (list(map(lambda x: ', зарплата ' + x + ' руб', ''.join(str(list(salaries)))[1:-1].split(', '))))))

    return [''.join(i) for i in res]

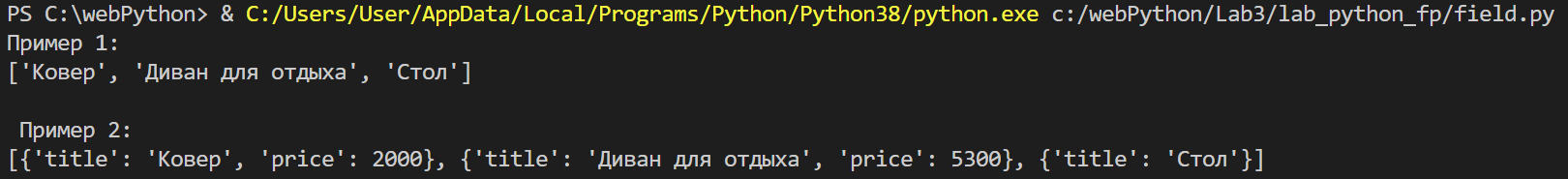
if \_\_name\_\_ == '\_\_main\_\_':

    with cm\_timer\_1():

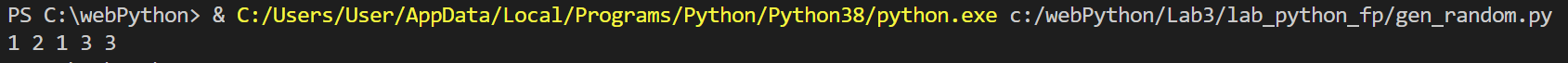
        f4(f3(f2(f1(data))))

***Экранные формы с примерами выполнения программы***

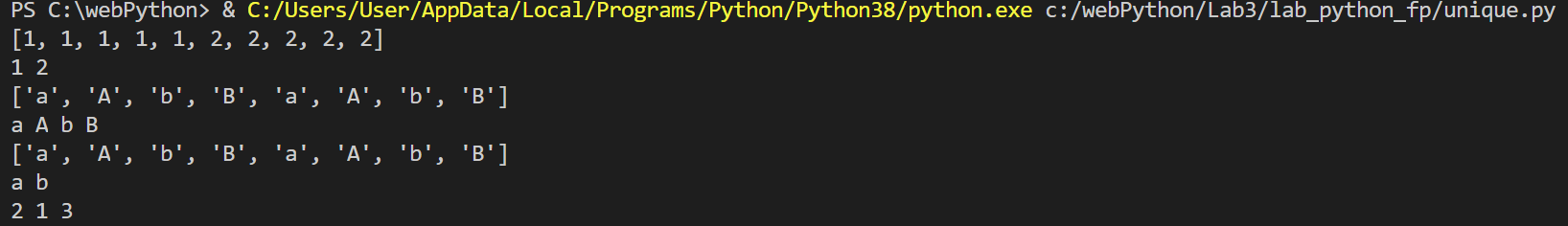
*Задача №1*

**

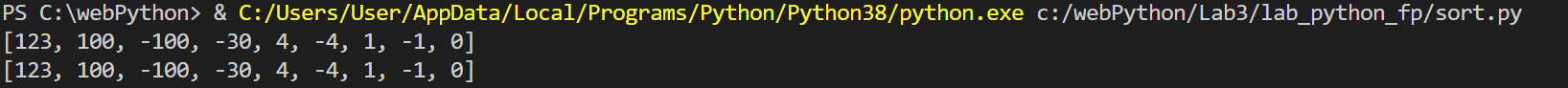
*Задача №2*

**

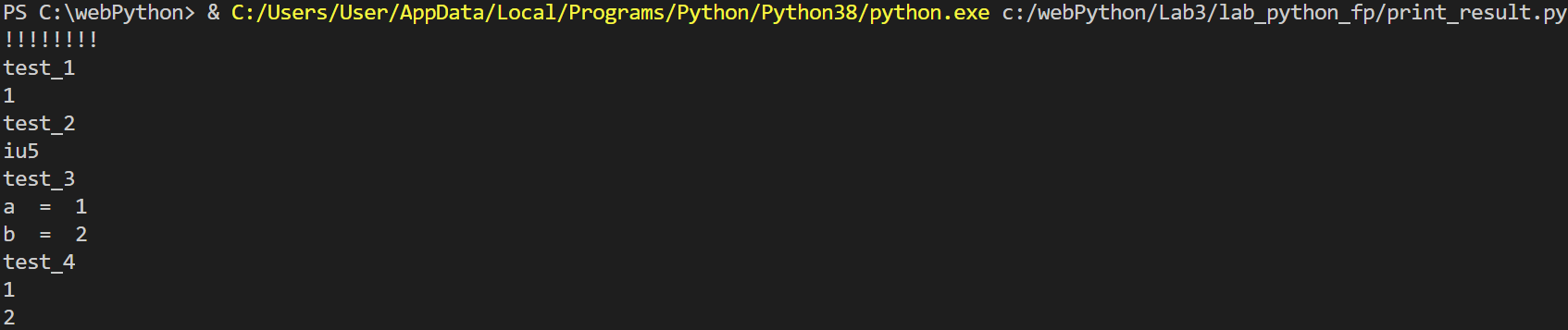
*Задача №3*

**

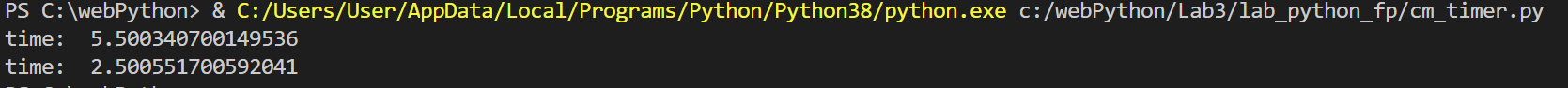
*Задача №4*

**

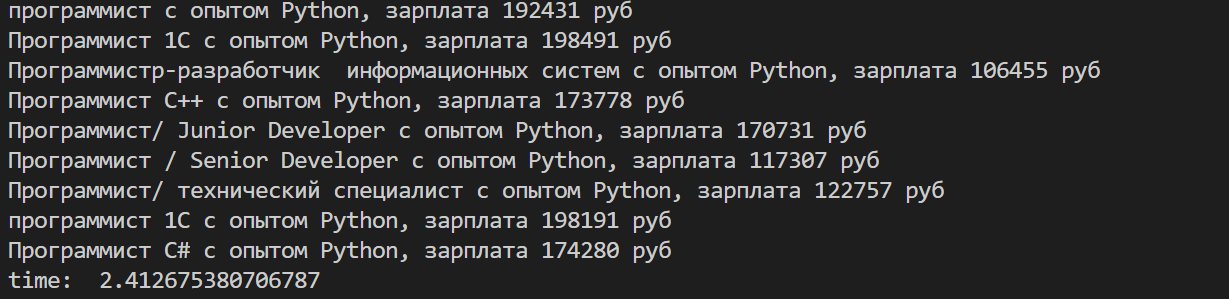
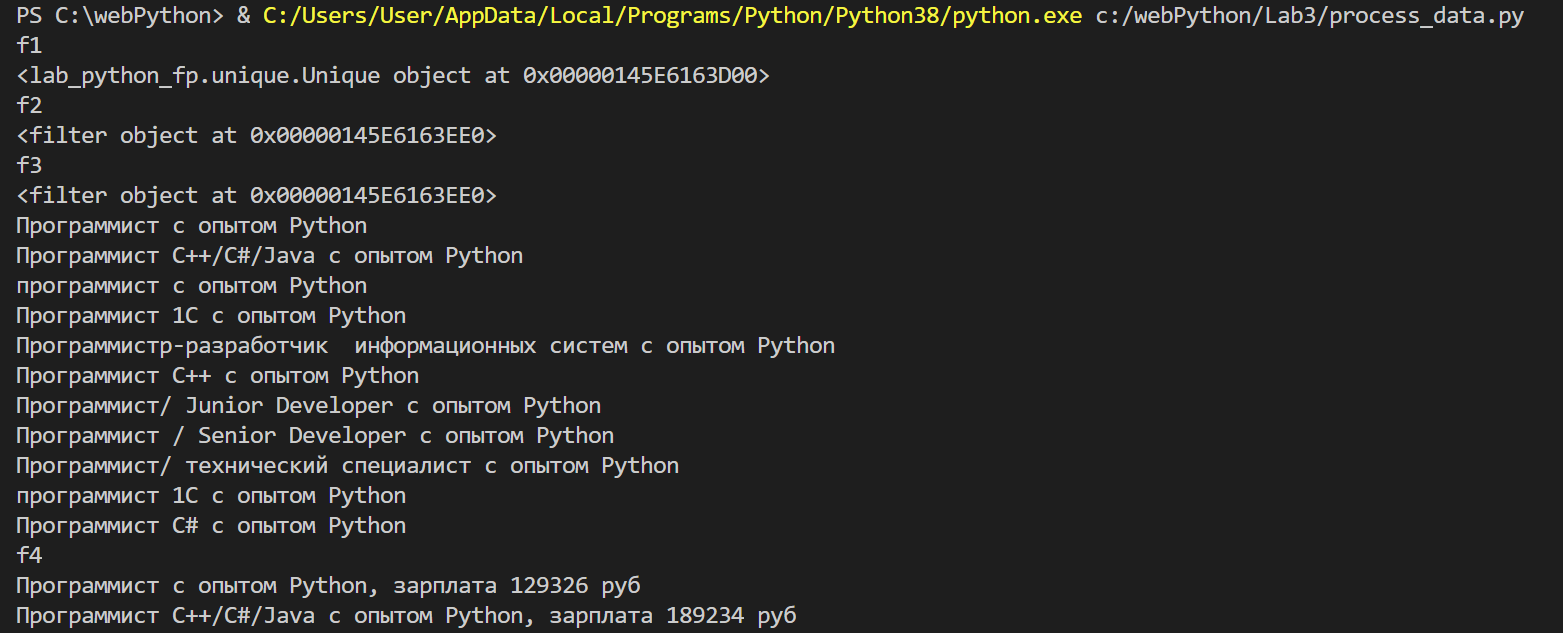
*Задача №5*

**

*Задача №6*

**

*Задача №7*

**