**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Разработка Интернет-приложений»

Отчет по лабораторной работе №2

«Python. Функциональные возможности»

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студентка группы ИУ5-53 |  | преподаватель каф. ИУ5 |
| Беспалова У.А. |  | Гапанюк Ю.Е. |
| Подпись и дата:  22.09.2019 |  | Подпись и дата: |

Москва, 2019 г.

1. **Содержание задания**

Задача 1

Необходимо реализовать генераторы field и gen\_random

1. В качестве первого аргумента генератор принимает list, дальше через \*args генератор принимает неограниченное кол-во аргументов.
2. Если передан один аргумент, генератор последовательно выдает только значения полей, если поле равно None, то элемент пропускается
3. Если передано несколько аргументов, то последовательно выдаются словари, если поле равно None, то оно пропускается, если все поля None, то пропускается целиком весь элемент

Генератор gen\_random последовательно выдает заданное количество случайных чисел в заданном диапазоне

В ex\_1.py нужно вывести на экран то, что они выдают, с помощью кода в *одну строку*

Генераторы должны располагаться в librip/gen.py

### Задача 2

Необходимо реализовать итератор, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты. Конструктор итератора также принимает на вход именной bool-параметр ignore\_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False. Итератор **не должен модифицировать** возвращаемые значения.

### Задача 3

Дан массив с положительными и отрицательными числами. Необходимо одной строкой вывести на экран массив, отсортированный по модулю. Сортировку осуществлять с помощью функции sorted

### Задача 4

Необходимо реализовать декоратор print\_result, который выводит на экран результат выполнения функции. Файл ex\_4.py **не нужно** изменять.

Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции, печатать результат и возвращать значение.

Если функция вернула список (list), то значения должны выводиться в столбик.

### Задача 5

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран

Задача 6

В репозитории находится файл data\_light.json. Он содержит облегченный список вакансий в России в формате json.

Структура данных представляет собой массив словарей с множеством полей: название работы, место, уровень зарплаты и т.д.

В ex\_6.py дано 4 функции. В конце каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print\_result печатается результат, а контекстный менеджер timer выводит время работы цепочки функций.

Задача реализовать все 4 функции по заданию, ничего не изменяя в файле-шаблоне. Функции f1-f3 должны быть реализованы в 1 строку, функция f4 может состоять максимум из 3 строк.

Что функции должны делать:

1. Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна **игнорировать регистр**. Используйте наработки из предыдущих заданий.
2. Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Иными словами нужно получить все специальности, связанные с программированием. Для фильтрации используйте функцию filter.
3. 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.

1. Текст программы
2. Модуль gens.py

import random

def field(items, \*args):

assert len(args) > 0

if len(args)==1:

for i in items:

if args[0] in i:

yield i[args[0]] #вернули генератор

else:

for i in items:

ivan = {one: i[one] for one in args if one in i}

if len(ivan)>0:

yield ivan

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

for i in range (num\_count):

yield random.randint (begin, end)

Работа модуля (ex\_1.py)

from librip.gens import field, gen\_random

goods = [

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

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

{'title': 'Стелаж', 'price': 7000, 'color': 'white'},

{'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}

]

print (list(gen\_random (2, 3, 10)))

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

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

1. Модуль Iterators.py

class Unique(object):

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

self.items = list(items)

self.map = []

self.i=0

if 'ignore\_case' not in kwargs:

self.ign = False

else:

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

def \_\_next\_\_(self):

if self.i == len(self.items):

raise StopIteration

result = self.items[self.i]

self.i += 1

if (self.ign==True) and (type(result) == str):

if str.casefold(result) not in self.map:

self.map.append(str.casefold(result))

return result

else:

return next(self)

elif result not in self.map:

self.map.append(result)

return result

else:

return next(self)

def \_\_iter\_\_(self):

return self

Работа модуля (ex\_2.py)

from librip.gens import gen\_random

from librip.iterators import Unique

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

data2 = gen\_random(1, 3, 10)

data3 = ['t', 'F', 'T', 'f']

print (list(Unique(data1)))

print (list(Unique(data2)))

print (list(Unique (data3, ignore\_case = False)))

print (list(Unique (data3, ignore\_case = True)))

1. Задание 3 (ex\_3.py)

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

print(sorted(data, key=lambda x: abs(x)))

1. Модуль decorators.py

def print\_result(func):

def fun(\*args):

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

result = func(\*args)

if isinstance(result, list): #если список

for item in result:

print(item)

elif isinstance(result, dict): #если словарь

for first, second in result.items():

print(first, "=", second)

else:

print(result)

return result

return fun

Работа модуля (ex\_4.py)

from librip.decorators import print\_result

@print\_result

def test\_1():

return 1

@print\_result

def test\_2():

return 'iu'

@print\_result

def test\_3():

return {'a': 1, 'b': 2}

@print\_result

def test\_4():

return [1, 2]

test\_1()

test\_2()

test\_3()

test\_4()

1. Модуль ctxmngrs.py

from contextlib import contextmanager

import time

@contextmanager

def timer():

start = time.time()

yield

print (time.time() - start)

Работа модуля (ex\_5.py)

from time import sleep

from librip.ctxmngrs import timer

with timer():

sleep(2.5)

Задание 6 (ex\_6.py)

#!/usr/bin/env python3

import json

import sys

from librip.ctxmngrs import timer

from librip.decorators import print\_result

from librip.gens import field, gen\_random

from librip.iterators import Unique as unique

path ="C:\\Users\\Hi\\lab\_2\_Python\\data\_light\_cp1251.json"

with open(path) as f:

data = json.load(f)

@print\_result

def f1(arg):

return sorted(list(unique(field(arg, "job-name"), ignore\_case=True)), key=lambda x: str.casefold(x))

@print\_result

def f2(arg):

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

@print\_result

def f3(arg):

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

@print\_result

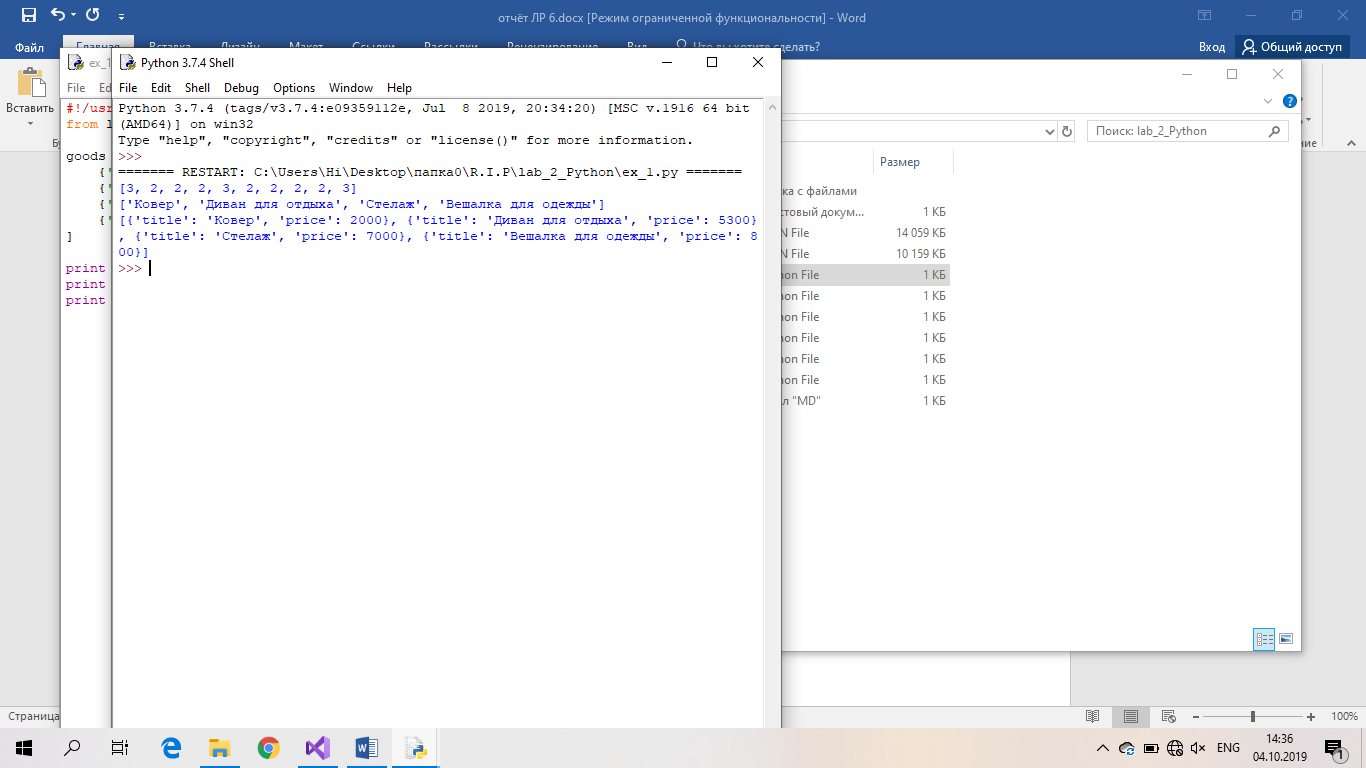
def f4(arg):

return dict(zip(arg, gen\_random(100000, 200000, len(arg))))

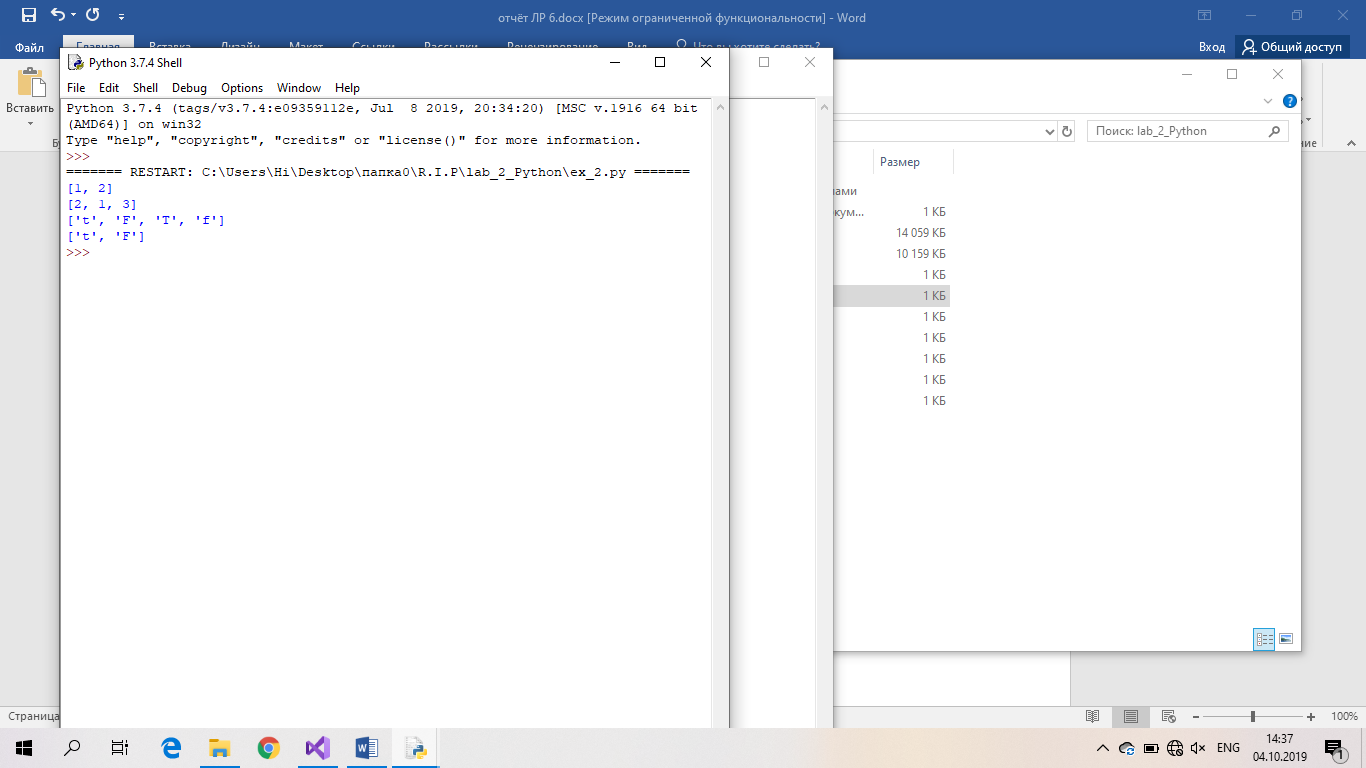
with timer():

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

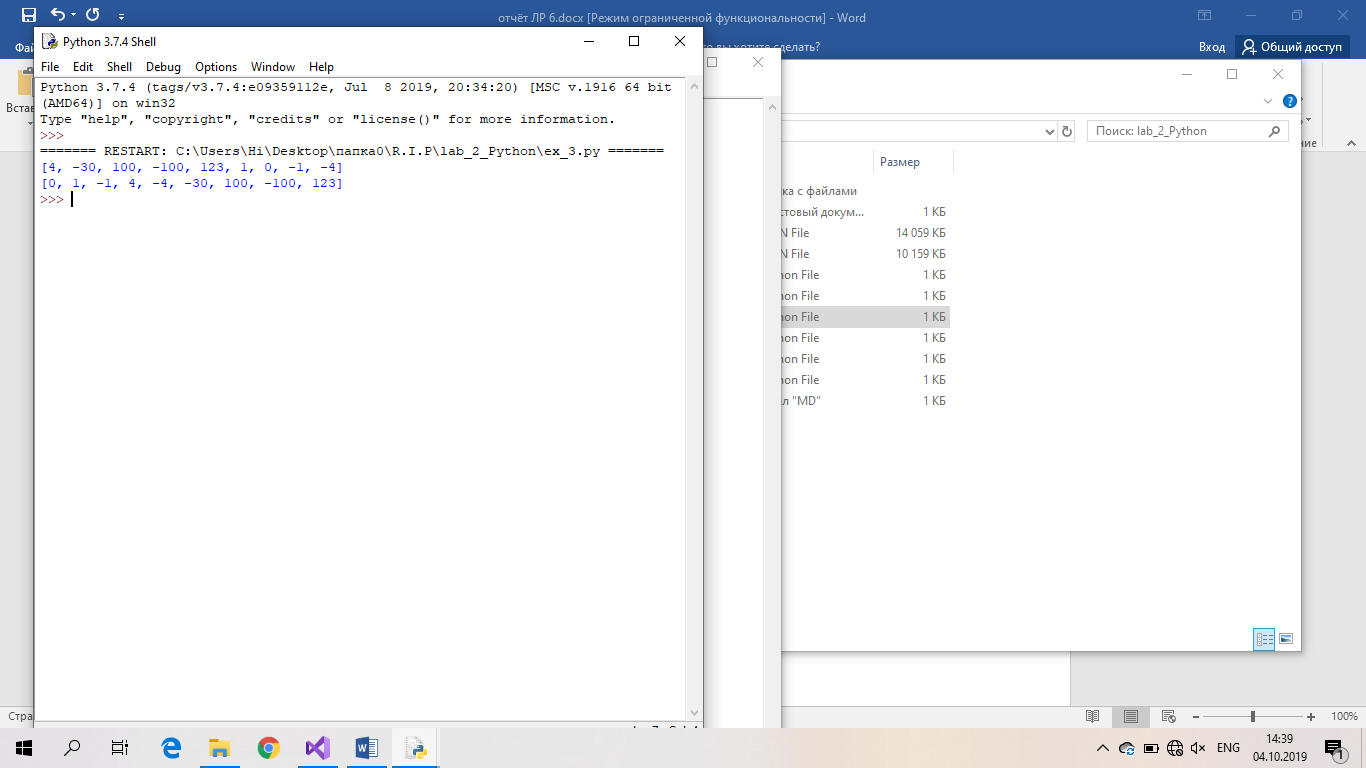
1. **Экранные формы с примерами работы программы**

ex\_1.py

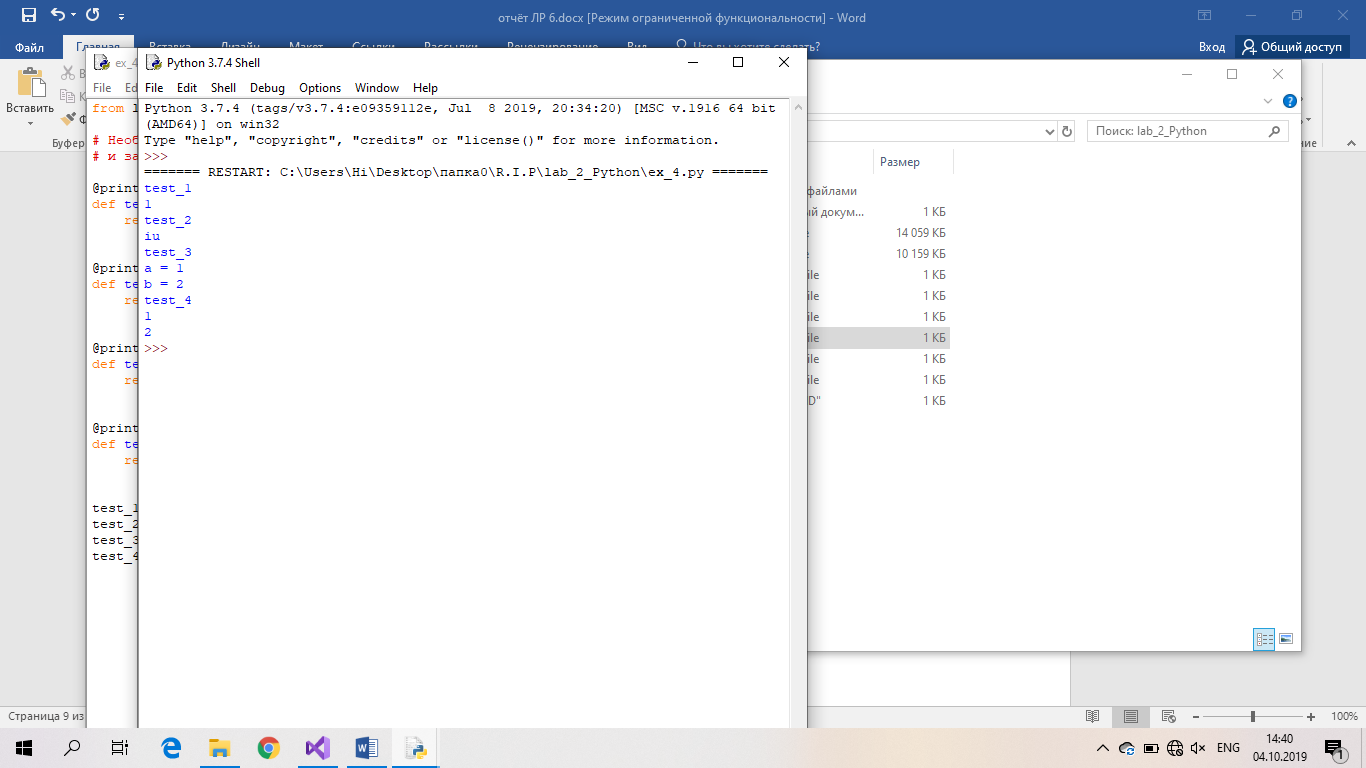
ex\_2.py



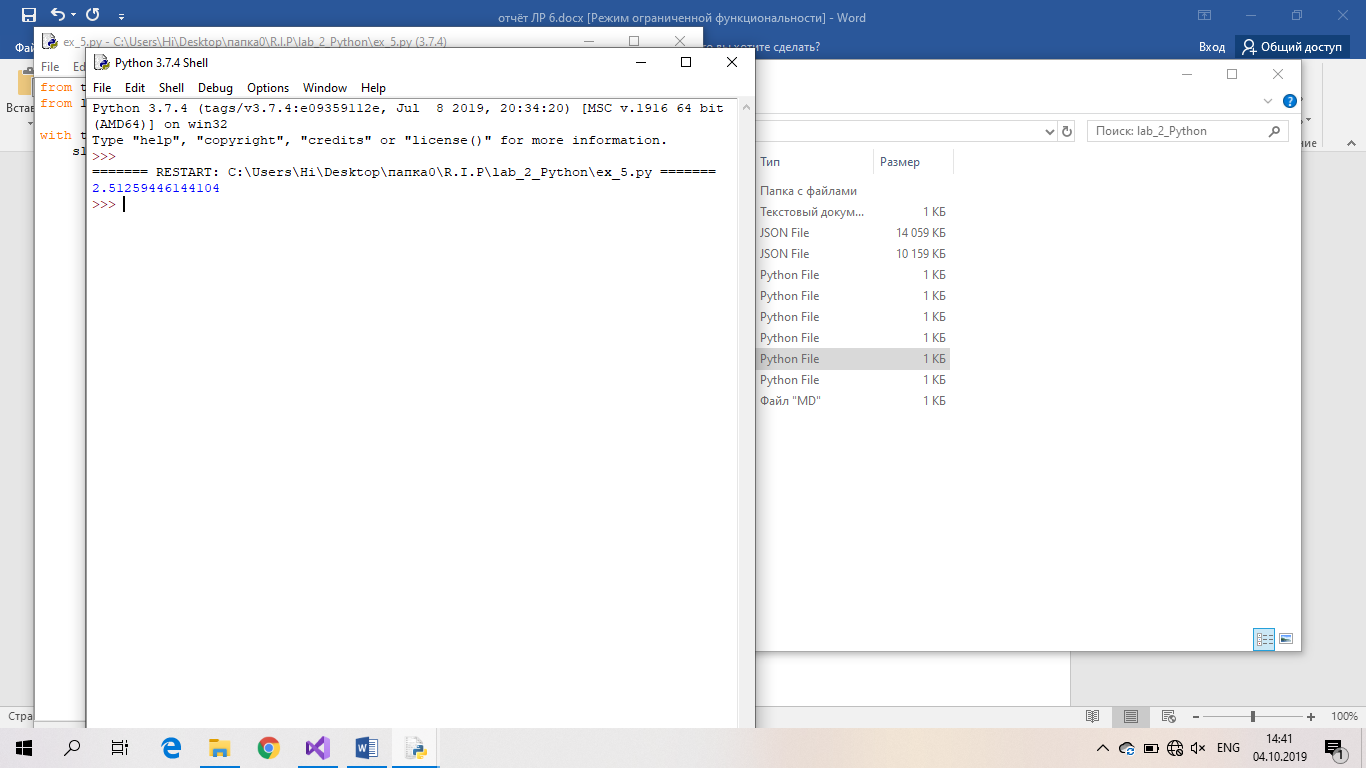
ex\_3.py



ex\_4.py



ex\_5.py



ex\_6.py

