Отчет по лабораторной работе №4

по курсу «Разработка интернет-приложений»

«Python. Функциональные возможности»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Преподаватель: |
| Сергеев Никита РТ5-51 |  | Гапанюк Ю.Е. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |

2016 г.

1. Задание лабораторной работы.

1. Зайти на github.com и выполнить ​ fork​ проекта с заготовленной структурой ​ https://github.com/iu5team/ex-lab4

2. Переименовать репозиторий в ​ lab\_4

3. Выполнить ​ git clone ​ проекта из вашего репозитория

4. *Задача 1 (​ ex\_1.py​ )*

Необходимо реализовать генераторы ​ field​ и ​ gen\_random

Генератор ​ field​ последовательно выдает значения ключей словарей массива

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

5. *Задача 2 (​ ex\_2.py​ )*

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.

6. *Задача 3 (​ ex\_3.py​ )*

Дан массив с положительными и отрицательными числами. Необходимо ​ одной строкой​ вывести на экран массив,

отсортированный по модулю. Сортировку осуществлять с помощью функции ​ sorted.

7. *Задача 4 (​ ex\_4.py​ )*

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. Если функция вернула словарь (​ dict​ ), то ключи и значения должны выводить в столбик через знак равно.

8. *Задача 5 (​ ex\_5.py​ )*

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

9. *Задача 6 (​ ex\_6.py​ )*

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:

* Функция ​ f1​ должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр.
* Функция ​ f2 ​ должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”.
* Функция ​ f3 ​ должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python).
* Функция ​ f4 ​ должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

2) Код

**gen.py**

from random import randint

**def** field(items, \*args):  
 **assert** len(args) > 0  
 *# Необходимо реализовать генератор* **for** item **in** items:  
 r = {}  
 **for** arg **in** args:  
 **if** arg **in** item:  
 **if** len(args) == 1 **and** item[arg] != **None**:  
 **yield** item[arg]  
 **else**:  
 **if** item[arg] **is not None**:  
 r[arg] = item[arg]  
 **if** len(r) > 0 **and** len(args) > 1:  
 **yield** r

**ex\_1.py**

*#!/usr/bin/env python3***from** librip.gen **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'**}  
]  
  
*# Реализация задания 1*print(list(field(goods, **'title'**)))  
print(list(field(goods, **'title'**, **'price'**)))  
print(list(gen\_random(0, 16, 10)))

**iterators.py**

**class** Unique(object):  
  
 **def** \_\_init\_\_(self, items, \*\*kwargs):  
 *# Нужно реализовать конструктор  
 # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore\_case,  
 # в зависимости от значения которого будут считаться одинаковые строки в разном регистре  
 # Например: ignore\_case = True, Aбв и АБВ разные строки  
 # ignore\_case = False, Aбв и АБВ одинаковые строки, одна из них удалится  
 # По-умолчанию ignore\_case = False* self.IGNORE\_CASE = kwargs[**'ignore\_case'**] **if 'ignore\_case' in** kwargs.keys() **else False** self.ITEMS = list(items)  
 self.PASSED = set()  
  
 **def** \_\_next\_\_(self):  
 *# Нужно реализовать \_\_next\_\_* **while True**:  
 **if** self.INDEX == len(self.ITEMS) - 1:  
 **raise** StopIteration  
 self.INDEX += 1  
 val = str(self.ITEMS[self.INDEX])  
 val2 = val **if** self.IGNORE\_CASE **else** val.lower()  
 **if** val2 **not in** self.PASSED:  
 self.PASSED.add(val2)  
 **return** val  
  
 **def** \_\_iter\_\_(self):  
 self.INDEX = -1  
 **return** self

**ex\_2.py**

*#!/usr/bin/env python3***from** librip.gen **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)  
  
*# Реализация задания 2*print(list(Unique(data1)))  
print(list(Unique(data2)))  
data = [**'a'**, **'A'**, **'b'**, **'B'**]  
print(list(Unique(data)))  
data = [**'a'**, **'A'**, **'b'**, **'B'**]  
print(list(Unique(data, ignore\_case=**True**)))

**ex\_3.py**

*#!/usr/bin/env python3*data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
*# Реализация задания 3*print(data)  
print(sorted(data, key=**lambda** key: abs(key)))

**ex\_4.py**

**from** librip.decorators **import** print\_result  
  
*# Необходимо верно реализовать print\_result  
# и задание будет выполнено*@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()

**ctxmngrs.py**

*# Здесь необходимо реализовать  
# контекстный менеджер timer  
# Он не принимает аргументов, после выполнения блока он должен вывести время выполнения в секундах  
# Пример использования  
# with timer():  
# sleep(5.5)  
#  
# После завершения блока должно вывестись в консоль примерно 5.5***import** time  
  
  
**class** timer:  
 **def** \_\_enter\_\_(self):  
 self.start = time.time()  
  
 **def** \_\_exit\_\_(self, exp\_type, exp\_value, traceback):  
 print(**"Execution time: "** + str(time.time() - self.start))

**ex\_5.py**

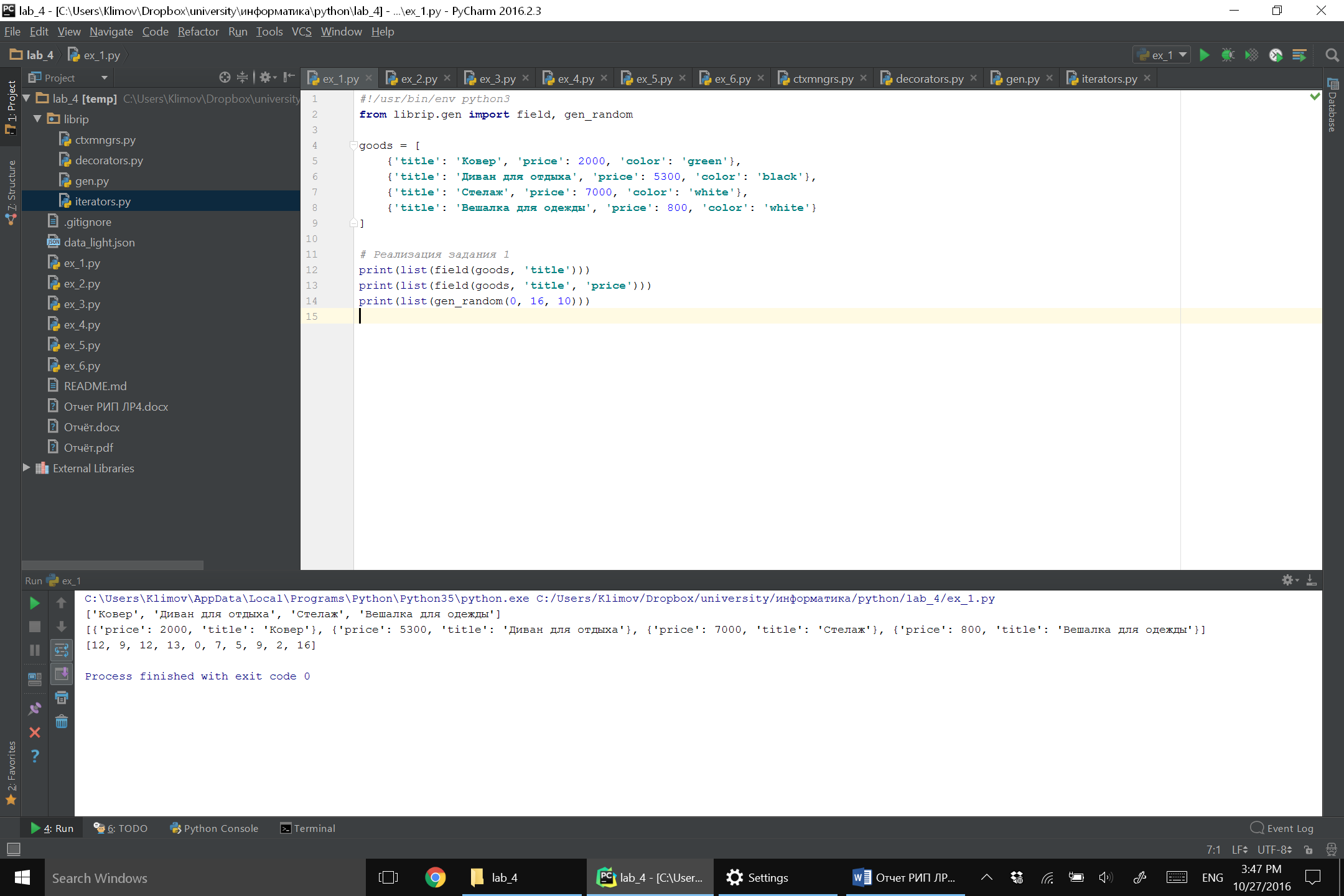
**from** time **import** sleep  
**from** librip.ctxmngrs **import** timer  
  
**with** timer():  
 sleep(5.5)

**ex\_6.py**

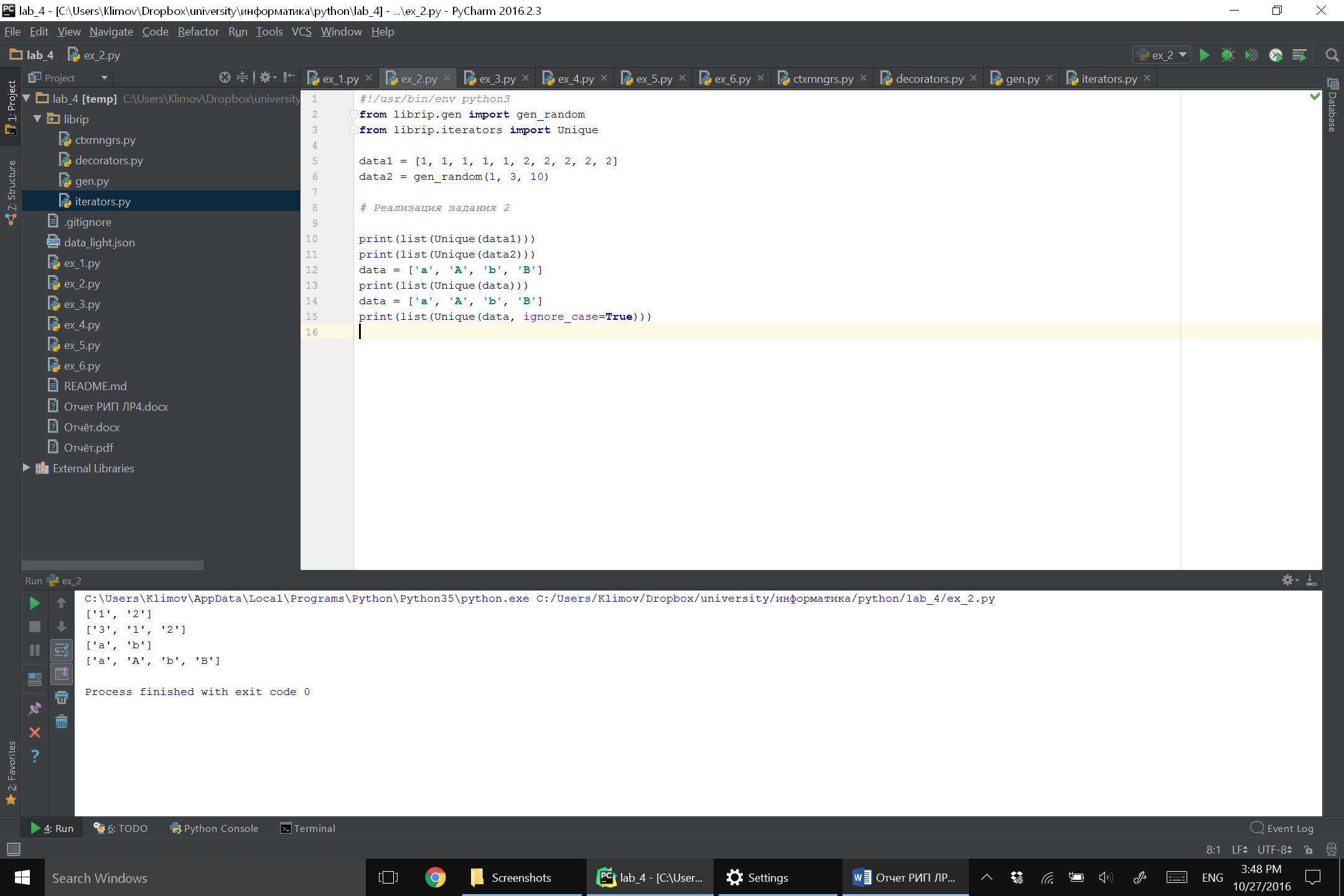
*#!/usr/bin/env python3***import** json  
**import** sys  
**from** librip.ctxmngrs **import** timer  
**from** librip.decorators **import** print\_result  
**from** librip.gen **import** field, gen\_random  
**from** librip.iterators **import** Unique **as** unique  
  
*# Здесь необходимо в переменную path получить  
# путь до файла, который был передан при запуске  
  
#assert len(sys.argv) > 0  
#path = sys.argv[0]*path = **"data\_light.json"  
  
with** open(path, encoding=**"utf8"**) **as** f:  
 data = json.load(f)  
  
  
*# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`  
# Важно!  
# Функции с 1 по 3 дожны быть реализованы в одну строку  
# В реализации функции 4 может быть до 3 строк  
# При этом строки должны быть не длиннее 80 символов*@print\_result  
**def** f1(arg):  
 **return** list(unique(list(field(arg, **"job-name"**)), ignore\_case=**True**))  
  
  
@print\_result  
**def** f2(arg):  
 **return** list(filter(**lambda** \_: **"Программист" in** \_, arg))  
  
  
@print\_result  
**def** f3(arg):  
 **return** list(map(**lambda** x: x + **" с опытом Python"**, arg))  
  
  
@print\_result  
**def** f4(arg):  
 **return** list(map(**lambda** x: **"{}, зарплата {} руб."**.format(x[0], x[1]),  
 zip(arg, gen\_random(100000, 200000, len(arg)))))  
  
**with** timer():  
 f4(f3(f2(f1(data))))

1. Результаты работы

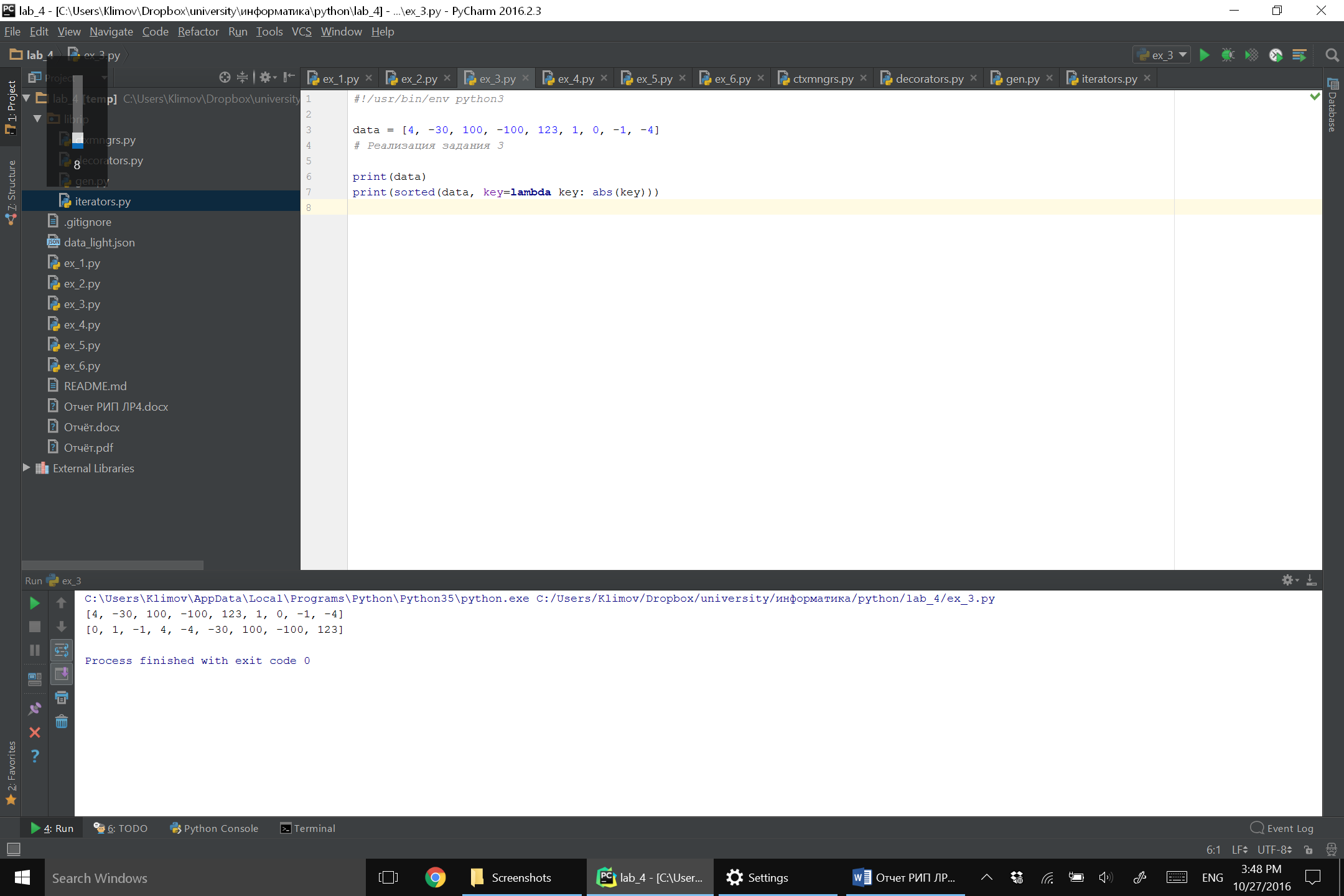
*Задача 1*

**

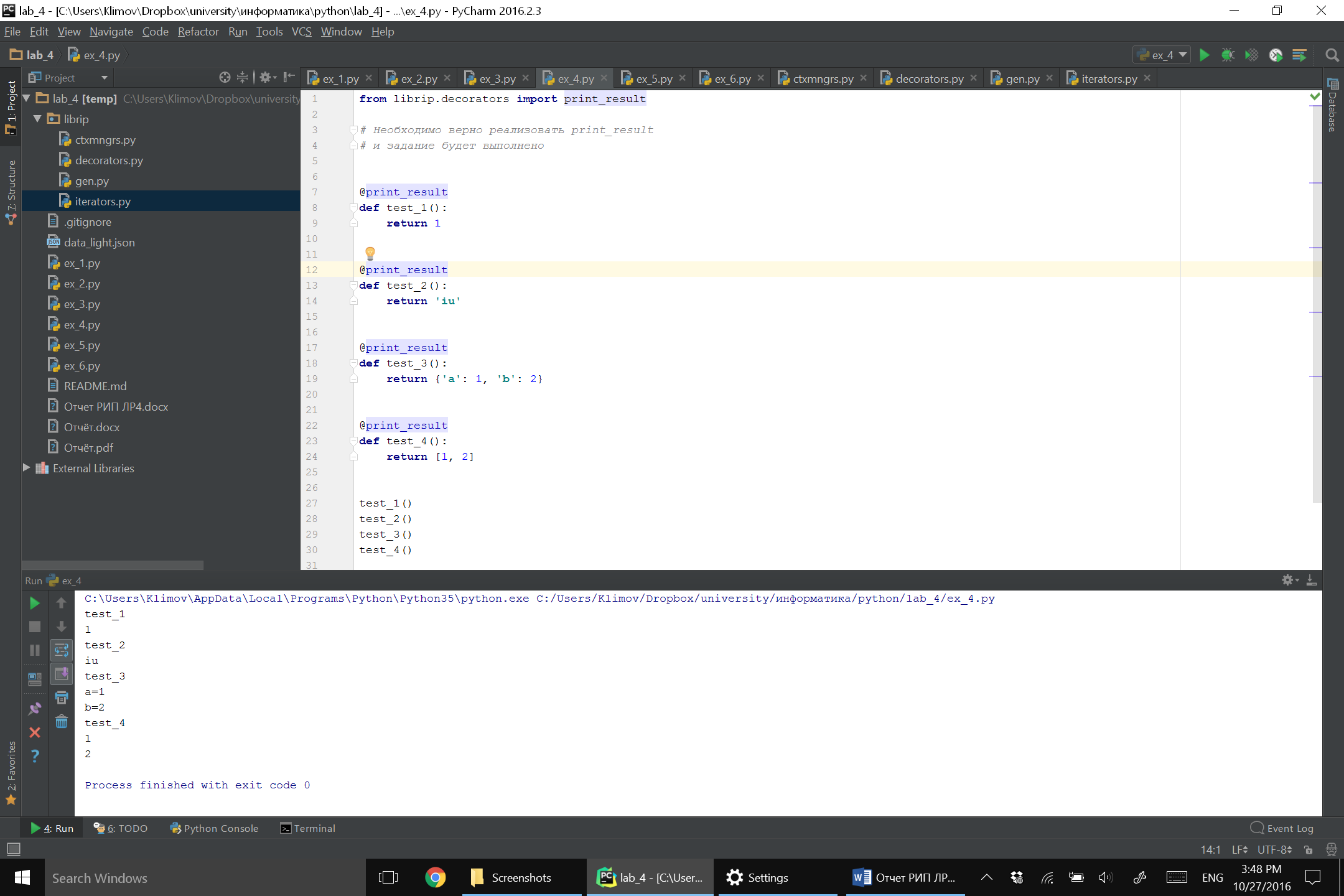
*Задача 2*



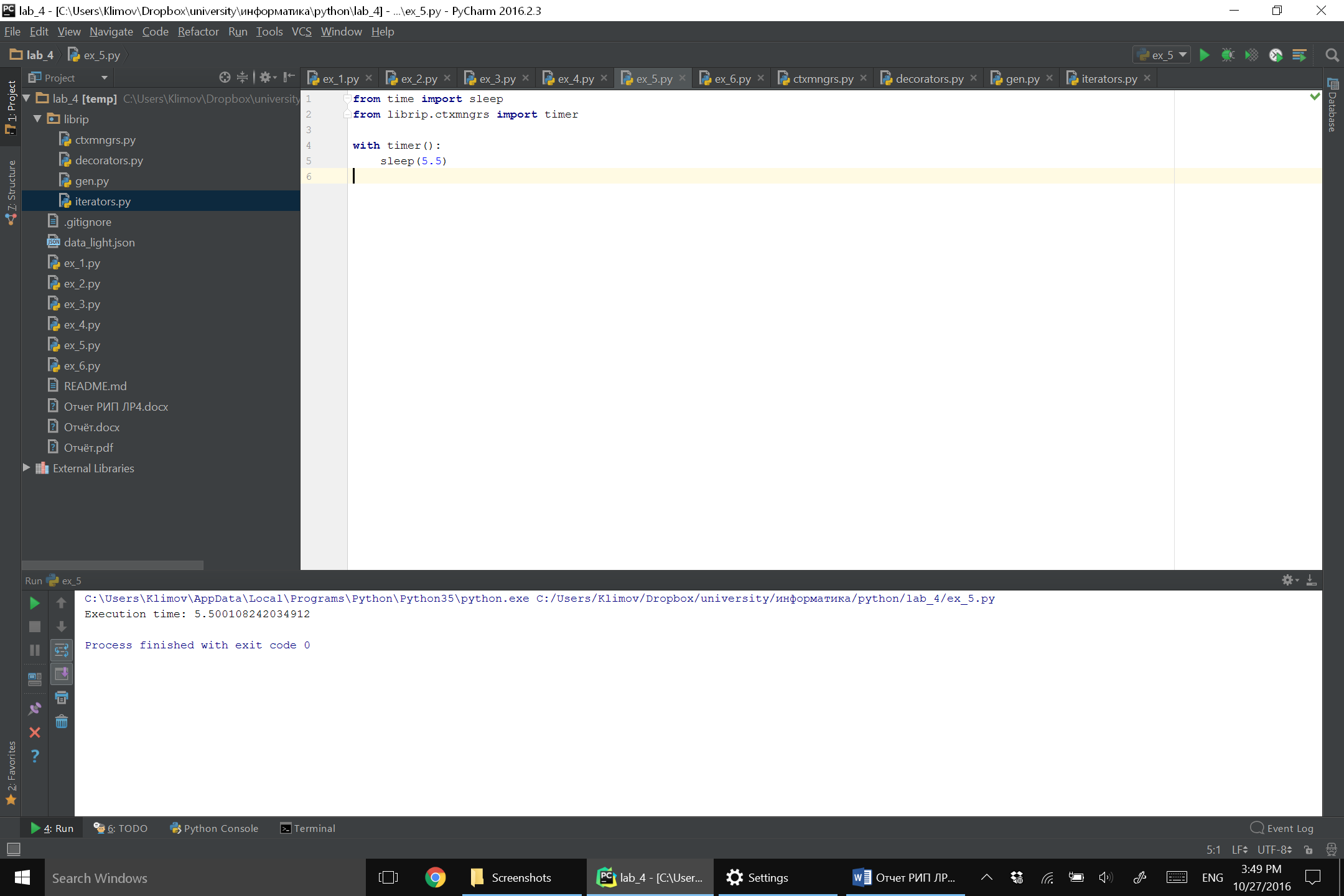
*Задача 3*



*Задача 4*



*Задача 5*



*Задача 6*

