

МГТУ им. Н. Э. Баумана, кафедра ИУ5  
курс “Разработка интернет-приложений”

Лабораторная работа №2  
Python. Функциональные возможности

ВЫПОЛНИЛ:

Матюнин да Вейга Р.А.

Группа: ИУ5-51Б

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2019

## Задание и порядок выполнения

**Важно** выполнять все задачи последовательно. С 1 по 5 задачу формируется модуль `librip`, с помощью которого будет выполняться задание 6 на реальных данных из жизни. Весь вывод на экран (даже в столбик) необходимо запрограммировать одной строкой.

### Подготовительный этап

1. Зайти на `github.com` и выполнить `fork` проекта с заготовленной структурой `https://github.com/iu5team/ex-lab4`
2. Переименовать репозиторий в `lab_2`
3. Выполнить `git clone` проекта из вашего репозитория

### Задача 1 (`ex_1.py`)

Необходимо реализовать генераторы `field` и `gen_random`

Генератор `field` последовательно выдает значения ключей словарей массива

Пример:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]
```

`field(goods, 'title')` должен выдавать 'Ковер', 'Диван для отдыха'

`field(goods, 'title', 'price')` должен выдавать `{'title': 'Ковер', 'price': 2000}`, `{'title': 'Диван для отдыха'}`

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

Пример:

`gen_random(1, 3, 5)` должен выдать 5 чисел от 1 до 3, т.е. примерно 2, 2, 3, 2, 1

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 (`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`. Итератор **не должен модифицировать** возвращаемые значения.

Пример:

```
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

`Unique(data)` будет последовательно возвращать только 1 и 2

В `ex_2.py` нужно вывести на экран то, что они выдают *одной строкой*. **Важно** продемонстрировать работу как с массивами, так и с генераторами (`gen_random`).

Итератор должен располагаться в `librip/iterators.py`

### 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

Пример:

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

Вывод: [0, 1, -1, 4, -4, -30, 100, -100, 123]

### 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 выводиться в столбик через знак равно

Пример:

```
@print_result
```

```
def test_1():
```

```
    return 1
```

```
test_1()
```

На консоль выведется:

```
test_1
```

```
1
```

Декоратор должен располагаться в librip/decorators.py

### Задача 5 (ex\_5.py)

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

Пример:

```
with timer():
```

```
    sleep(5.5)
```

После завершения блока должно вывестись в консоль примерно 5.5

### Задача 6 (ex\_6.py)

Мы написали все инструменты для работы с данными. Применим их на реальном примере, который мог возникнуть в жизни. В репозитории находится файл data\_light.json. Он содержит облегченный список вакансий в России в формате json (ссылку на полную версию размером ~ 1 Гб. в формате xml можно найти в файле README.md).

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.

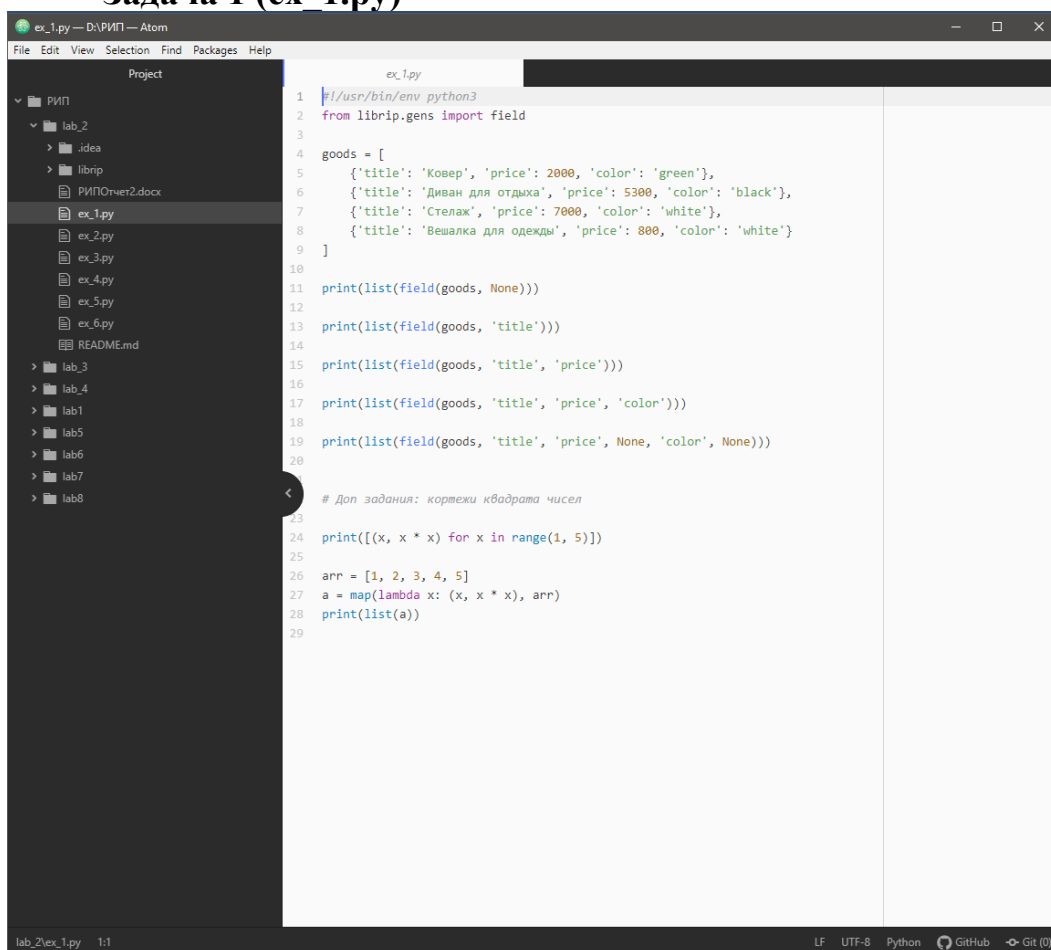
Что функции должны делать:

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`.
4. 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.

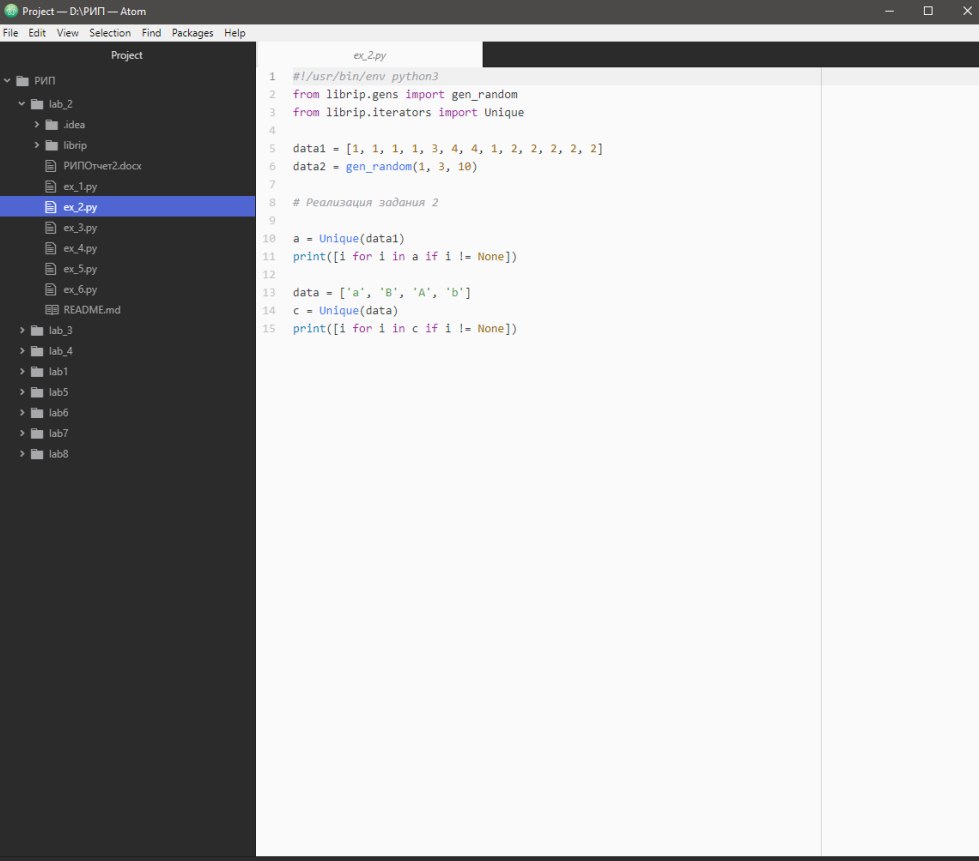
## Выполненная работа

### Задача 1 (ex\_1.py)



```
1 #!/usr/bin/env python3
2 from librip.gens import field
3
4 goods = [
5     {'title': 'Ковёр', 'price': 2000, 'color': 'green'},
6     {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
7     {'title': 'Стелаж', 'price': 7000, 'color': 'white'},
8     {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}
9 ]
10
11 print(list(field(goods, None)))
12
13 print(list(field(goods, 'title')))
14
15 print(list(field(goods, 'title', 'price')))
16
17 print(list(field(goods, 'title', 'price', 'color')))
18
19 print(list(field(goods, 'title', 'price', None, 'color', None)))
20
21 # Доп задания: кортежи квадрата чисел
22
23 print([(x, x * x) for x in range(1, 5)])
24
25
26 arr = [1, 2, 3, 4, 5]
27 a = map(lambda x: (x, x * x), arr)
28 print(list(a))
29
```

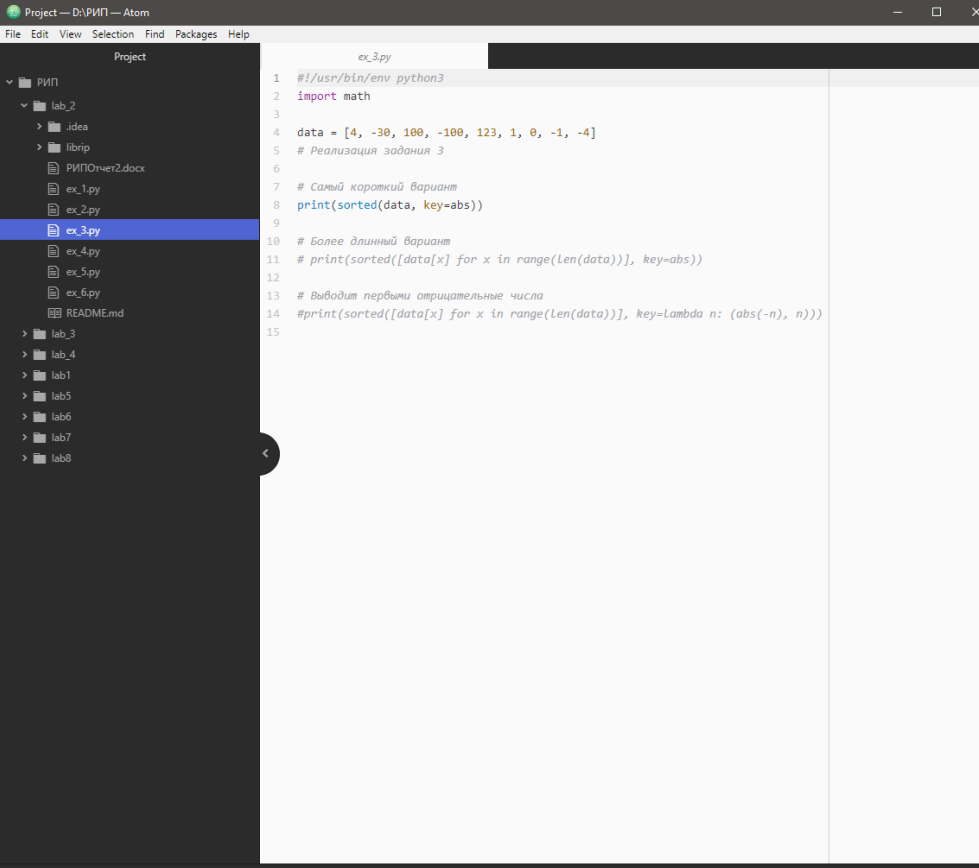
## Задача 2 (ex\_2.py)



```
1 #!/usr/bin/env python3
2 from librip.gens import gen_random
3 from librip.iterators import Unique
4
5 data1 = [1, 1, 1, 1, 3, 4, 4, 1, 2, 2, 2, 2]
6 data2 = gen_random(1, 3, 10)
7
8 # Реализация задания 2
9
10 a = Unique(data1)
11 print([i for i in a if i != None])
12
13 data = ['a', 'B', 'A', 'b']
14 c = Unique(data)
15 print([i for i in c if i != None])
```

lab\_2\ex\_2.py 1:1 LF UTF-8 Python GitHub Git (0)

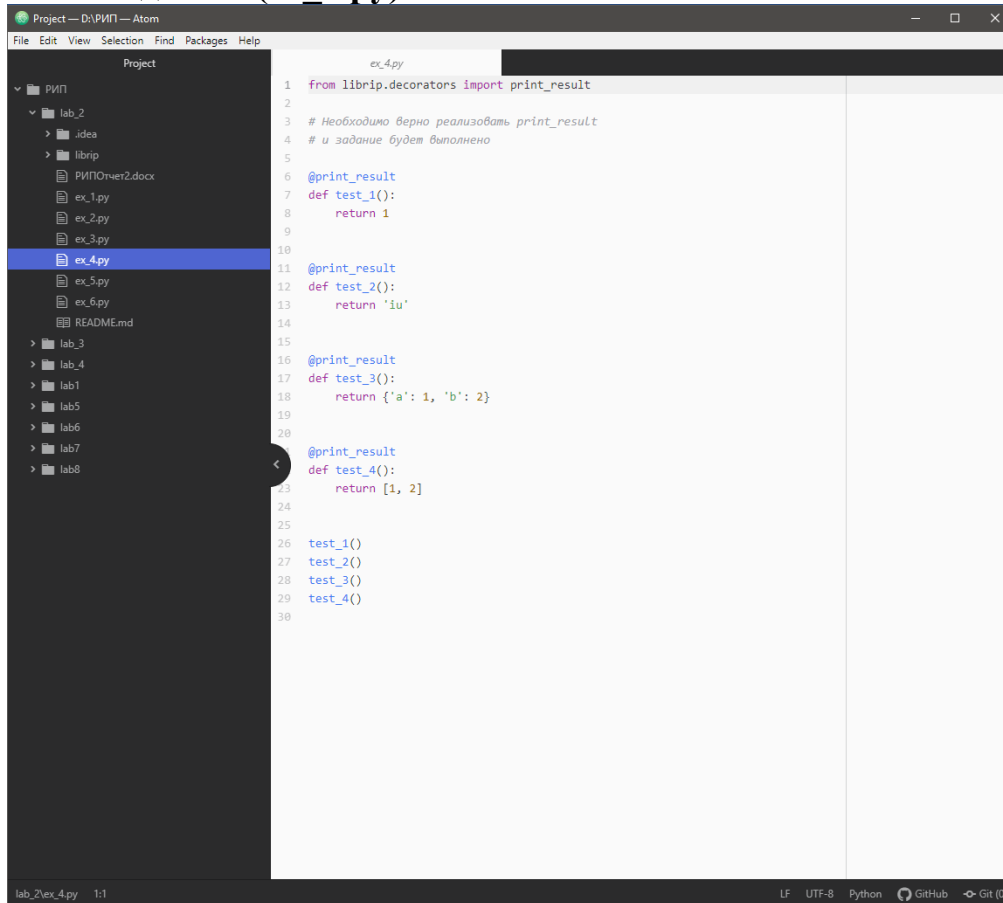
## Задача 3 (ex\_3.py)



```
1 #!/usr/bin/env python3
2 import math
3
4 data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
5 # Реализация задания 3
6
7 # Самый короткий вариант
8 print(sorted(data, key=abs))
9
10 # Более длинный вариант
11 # print(sorted([data[x] for x in range(len(data))], key=abs))
12
13 # Выводит первыми отрицательные числа
14 # print(sorted([data[x] for x in range(len(data))], key=lambda n: (abs(-n), n)))
15
```

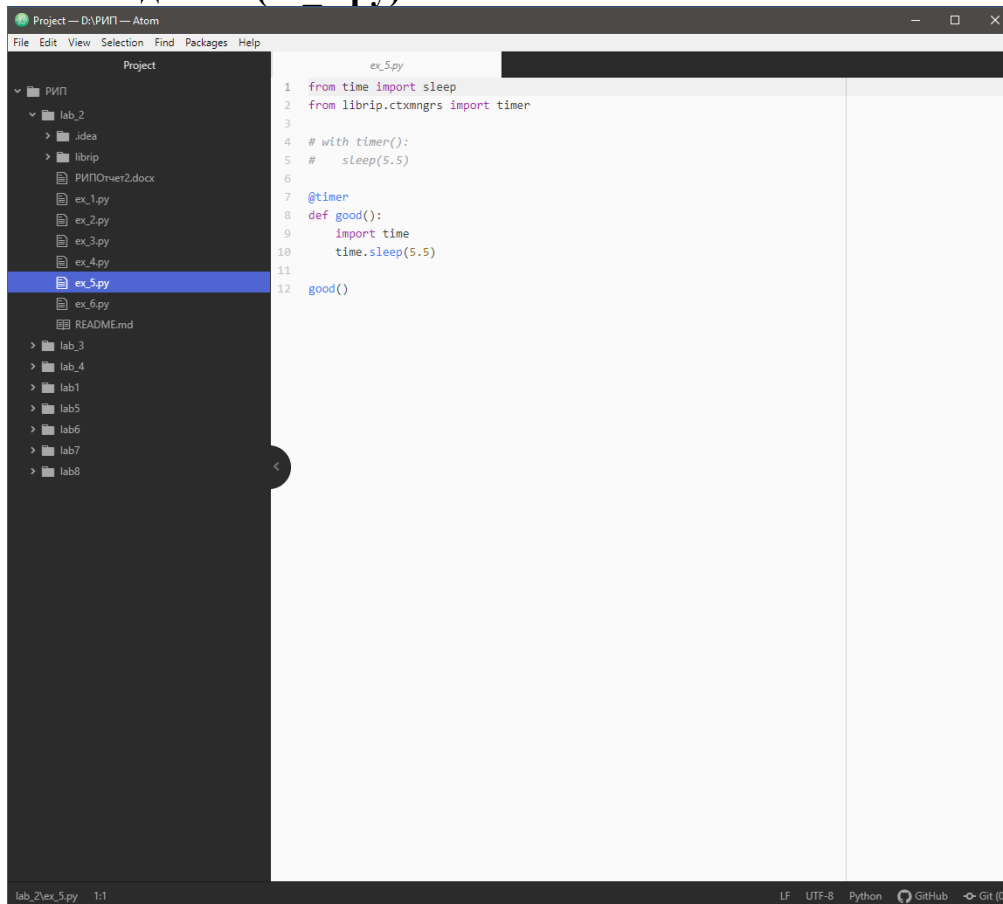
lab\_2\ex\_3.py 1:1 LF UTF-8 Python GitHub Git (0)

## Задача 4 (ex\_4.py)



```
ex_4.py
1 from librip.decorators import print_result
2
3 # Необходимо верно реализовать print_result
4 # и задание будет выполнено
5
6 @print_result
7 def test_1():
8     return 1
9
10
11 @print_result
12 def test_2():
13     return 'iu'
14
15
16 @print_result
17 def test_3():
18     return {'a': 1, 'b': 2}
19
20
21 @print_result
22 def test_4():
23     return [1, 2]
24
25
26 test_1()
27 test_2()
28 test_3()
29 test_4()
30
```

## Задача 5 (ex\_5.py)



```
ex_5.py
1 from time import sleep
2 from librip.ctxmngns import timer
3
4 # with timer():
5 #     sleep(5.5)
6
7 @timer
8 def good():
9     import time
10    time.sleep(5.5)
11
12 good()
```

## Задача 6 (ex\_6.py)

Project — D:\РИП — Atom

File Edit View Selection Find Packages Help

Project

- РИП
  - lab\_2
    - idea
    - librip
    - РИПОчмет2.docx
    - ex\_1.py
    - ex\_2.py
    - ex\_3.py
    - ex\_4.py
    - ex\_5.py
    - ex\_6.py**
    - README.md
  - lab\_3
  - lab\_4
  - lab1
  - lab5
  - lab6
  - lab7
  - lab8

```
ex_6.py
1 #!/usr/bin/env python3
2 import json
3 import sys
4 from librip.ctxmgrs import timer
5 from librip.decorators import print_result
6 from librip.gens import field, gen_random
7 from librip.iterators import Unique as unique
8
9 path = 'data_light.json'
10
11 # Здесь необходимо в переменную path получить
12 # путь до файла, который был передан при запуске
13
14 with open(path) as f:
15     data = json.load(f)
16
17 # Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
18 # Важно!
19 # Функции с 1 по 3 должны быть реализованы в одну строку
20 # В реализации функции 4 может быть до 3 строк
21 # При этом строки должны быть не длиннее 80 символов
22
23 @print_result
24 def f1():
25     return sorted(list(field(data, 'job-name')))
26     # raise NotImplemented
27
28 # Фильтр: выводит слова, в которых есть слово 'программист'
29 def prog(arr):
30     if ('программист' in arr):
31         return arr
32
33 @print_result
34 def f2():
35     return list(filter(prog, f1()))
36     #raise NotImplemented
37
38 def python(arr):
39     return str(arr) + ' с опытом Python'
40
41 @print_result
42 def f3():
43     return list(map(python, f2()))
44     #raise NotImplemented
```

lab\_2\ex\_6.py 1:1

LF UTF-8 Python GitHub Git (0)

Project — D:\РИП — Atom

File Edit View Selection Find Packages Help

Project

- РИП
  - lab\_2
    - idea
    - librip
    - РИПОчмет2.docx
    - ex\_1.py
    - ex\_2.py
    - ex\_3.py
    - ex\_4.py
    - ex\_5.py
    - ex\_6.py**
    - README.md
  - lab\_3
  - lab\_4
  - lab1
  - lab5
  - lab6
  - lab7
  - lab8

```
ex_6.py
37
38 def python(arr):
39     return str(arr) + ' с опытом Python'
40
41 @print_result
42 def f3():
43     return list(map(python, f2()))
44     #raise NotImplemented
45
46
47 @print_result
48 def f4():
49     return list(zip(f3(), gen_random(100000, 200000, len(f3()))))
50     #raise NotImplemented
51
52 # @timer
53 # def good():
54 #     f4(f3(f2(f1())))
55 #
56 # good()
57 with timer():
58     f4(f3(f2(f1(data))))
59
60
61
62 #####_декоратор_печатающий_результат_функции_#####
63
64 # def print_result(func):
65 #     def wrapper(*args, **kwargs):
66 #         rc = func(*args, **kwargs)
67 #         print('function ' + func.__name__ + ' returns ' + repr(rc))
68 #         return rc
69 #     return wrapper
70 #
71 #
72 # @print_result
73 # def a(x=0):
74 #     return x, x*x
75 #
76 # print('a(2) =', a(2))
77
78 #####
79
```

lab\_2\ex\_6.py 1:1

LF UTF-8 Python GitHub Git (0)



# Результаты

```
Командная строка
C:\Users\R A T I\lab_2>python ex_1.py
[]
['Ковер', 'Диван для отдыха', 'Стелаж', 'Вешалка для одежды']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Стелаж', 'price': 7000}, {'title': 'Вешалка для одежды', 'price': 800}]
[{'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'}]
[{'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'}]
[(1, 1), (2, 4), (3, 9), (4, 16)]
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]

C:\Users\R A T I\lab_2>python ex_2.py
[1, 3, 4, 2]
['a', 'B', 'A', 'b']

C:\Users\R A T I\lab_2>python ex_3.py
[0, 1, -1, 4, -4, -30, 100, -100, 123]

C:\Users\R A T I\lab_2>python ex_4.py
test_1
1
test_2
iu
test_3
a = 1
b = 2
test_4
1
2

C:\Users\R A T I\lab_2>python ex_5.py
Function good : 5.500416994094849(sec)
```