**Московский государственный технический университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Разработка Интернет-приложений»

# Отчет по лабораторной работе №2
# «Python. Функциональные возможности»

Выполнил:
студент группы ИУ5-54Б
Коваленко Артём

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Подпись и дата:

Подпись и дата:

Москва, 2019 г.

**Описание задания лабораторной работы**

# Задание

**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.

## Задача 1 (`ex_1.py`)

Необходимо реализовать генераторы `field` и `gen_random`
Генератор `field` последовательно выдает значения ключей словарей массива

## 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.

## 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`

## 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

## Задача 5 (`ex_5.py`)

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

## 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.

---

## Source

- **ex_1.py**

```
from librip.gens import field
from librip.gens import 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'}
]
a = [1, 2, 3, 4, 5]
print(type(a))
[print('|', i, '|', end=" ") for i in field(goods, 'title')]
print()
[print('|', i, '|', end=" ") for i in field(goods, 'title',
'color')]
print()
[print('|', i, '|', end=" ") for i in gen_random(1, 3, 5)]
```

- **ex_2.py**

```
from librip.gens import gen_random
from librip.iterators import Unique
data1 = ['a', 'A', 'a', 1, 1, 2, 2, 2, 2, 2]
data2 = gen_random(1, 10, 10)
[print(i) for i in Unique(data1, ignore_case=True)]
```

```
print("_")
[print(i) for i in Unique(data1, ignore_case=False)]
print("_")
[print(i) for i in Unique(data2)]
```

- **ex_3.py**

```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
print(sorted(data, key=lambda x: abs(x)))
```

- **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()
```

- **ex_5.py**

```
from time import sleep
from librip.ctxmngrs import timer
with timer():
    sleep(2.5)
    sleep(3.5)
```

- **ex_6.py**

```
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 = sys.argv[1]
with open(path) as f:
```

```
    data = json.load(f)
@print_result
def f1(arg):
    return sorted(Unique(field(arg, 'job-name'),
ignore_case=True), key=lambda x: x.lower())
@print_result
def f2(arg):
    return list(filter(lambda x: (not
x.lower().find("программист")), arg))
@print_result
def f3(arg):
    return list(map(lambda x: (x + " с опытом Python"), arg))
@print_result
def f4(arg):
    return list(zip(arg,
                    map(lambda x: ("зарплата " + str(x) + "
руб."),
                        gen_random(100000, 200000, len(arg)))))
with timer():
    f4(f3(f2(f1(data))))
```

---

Скриншоты с результатами выполнения

```
>>> exec(open("ex_1.py").read())
<class 'list'>
| Ковер | | Диван для отдыха | | Стелаж | | Вешалка для одежды |
| {'title': 'Ковер', 'color': 'green'} | | {'title': 'Диван для отдыха', 'color': 'black'} | |
 {'title': 'Стелаж', 'color': 'white'} | | {'title': 'Вешалка для одежды', 'color': 'white'} |
| 1 | | 1 | | 1 | | 1 | | 3 |
>>>
>>> exec(open("ex_2.py").read())
a
1
2

_
a
A
1
2

_
10
7
6
>>> exec(open("ex_3.py").read())
[0, 1, -1, 4, -4, -30, 100, -100, 123]

>>>
```

---

```
>>> exec(open("ex_4.py").read())
test_1
1
test_2
iu
test_3
a = 1
b = 2
a = 1
b = 2
test_4
1
2
>>> exec(open("ex_5.py").read())
6.001464792000206

>>>
ex_6.py
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-эксперт
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
web-разработчик
Программист/ технический специалист с опытом Python
Программистр-разработчик  информационных систем с опытом Python
f4
('Программист с опытом Python', 'зарплата 139400 руб.')
('Программист / Senior Developer с опытом Python', 'зарплата 169907 руб.')
('Программист 1С с опытом Python', 'зарплата 190860 руб.')
('Программист C# с опытом Python', 'зарплата 160938 руб.')
('Программист C++ с опытом Python', 'зарплата 161845 руб.')
('Программист C++/C#/Java с опытом Python', 'зарплата 150437 руб.')
('Программист/ Junior Developer с опытом Python', 'зарплата 178858 руб.')
('Программист/ технический специалист с опытом Python', 'зарплата 136256 руб.')
('Программистр-разработчик  информационных систем с опытом Python', 'зарплата 171147 руб.')
0.03275460400000002

Process finished with exit code 0
```