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
Тарасов Владислав

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.

Исходный код

gens.py

```
import random

# Генератор вычленения полей из массива словарей
def field(items, * args):
  assert len(args) > 0
for item in items:
  if len(args) == 1:
  yield item[args[0]]

if len(args) != 1:
  res = {}
for key in args:
  res[key] = item[key]

yield res

# Генератор списка случайных чисел
def gen_random(begin, end, num_count):
  res = [random.randrange(begin, end + 1) for _ in range(num_count)]
return res
```

Iterators.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = items
        self.used = []
        self.index = 0
        self.ignore_case = kwargs.get('ignore_case')

    def __next__(self):
        while True:
            if self.index == len(self.items):
                raise StopIteration

            item_to_insert = self.items[self.index]

            if self.ignore_case:
                item_to_insert = self.items[self.index].lower()

            if item_to_insert not in self.used:
                self.used.append(item_to_insert)
                self.index += 1
                return self.items[self.index - 1]

            self.index += 1

    def __iter__(self):
        return self
```

ex_3.py

```
print(list(sorted(data, key=lambda num: abs(num))))
```

decorators.py

```
def print_result(f):
    def decorated_func(*args):
        print('Function name: ' + f.__name__)

        print('Function results:')
        print()
        t = f(*args)
        if type(t) is dict:
            for key in t.keys():
                print('{} = {}'.format(key, t[key]))
```

```
        else:
            if type(t) is list:
                for values in t:
                    print(values)
            else:
                print(t)
        print()
        return t

    return decorated_func
```

ctxmngrs.py

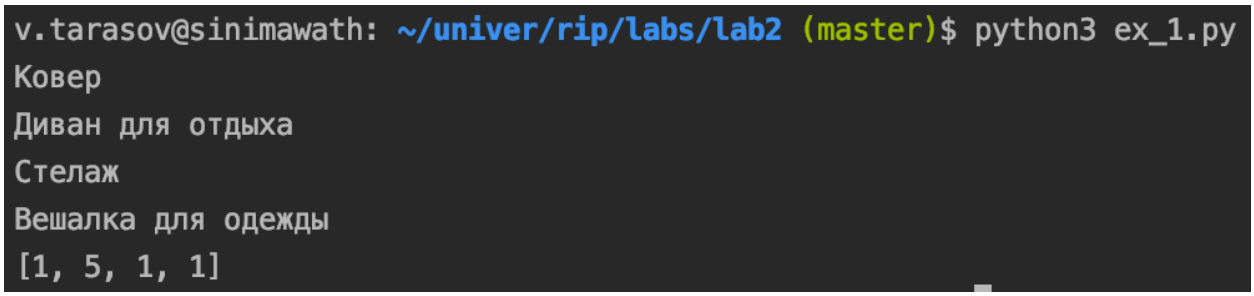
```
class timer:
    def __enter__(self):
        self.begin_time = time.time()

    def __exit__(self, type, value, traceback):
        print('Block has been executed during {:g} s'.format(time.time() - self.begin_time))
```

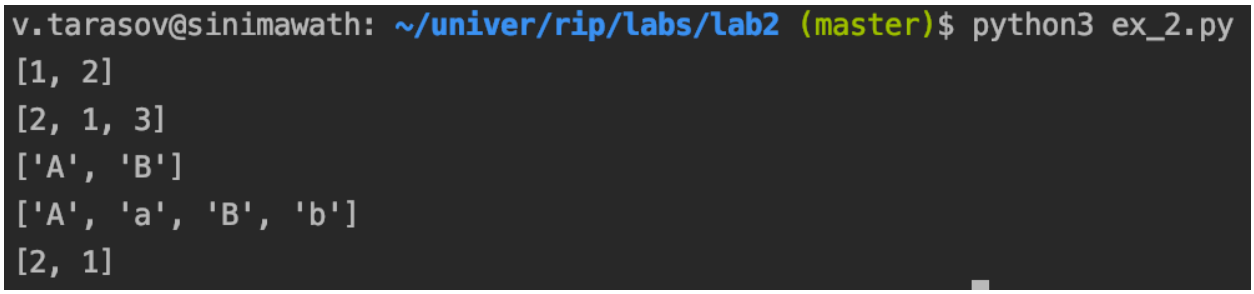
ex_6.py

```
# coding=utf-8
import json
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 = 'data_light.json'
with open(path) as f:
    data = json.load(f)


@print_result
def f1(arg):
    return sorted(unique(list(field(arg, 'job-name')), ignore_case=True))

@print_result
def f2(arg):
    return list(filter(lambda x: 'программист' in x, arg))


@print_result
def f3(arg):
    return list(map(lambda x: x + ' с опытом Python', arg))
```

```python
@print_result
def f4(arg):
    sals = list(gen_random(100000, 200000, len(arg)))
    return list(map(lambda x: x[1] + ' с зарплатой ' + str(sals[x[0]]), enumerate(arg)))


with timer():
    f4(f3(f2(f1(data))))
```

Скриншоты с результатами выполнения





```
v.tarasov@sinimawath: ~/univer/rip/labs/lab2 (master)$ python3 ex_4.py
Function name: test_1
Function results:

1

Function name: test_2
Function results:

iu

Function name: test_3
Function results:

a = 1
b = 2

Function name: test_4
Function results:

1
2

Process finished with exit code 0
```

```
v.tarasov@sinimawath: ~/univer/rip/labs/lab2 (master)$ python3 ex_5.py
Block has been executed during 1.50014 s
```

```
Function name: f4
Function results:

1С программист с опытом Python с зарплатой 172854
Web-программист с опытом Python с зарплатой 157080
Веб - программист (PHP, JS) / Web разработчик с опытом Python с зарплатой 174397
Веб-программист с опытом Python с зарплатой 110314
Ведущий инженер-программист с опытом Python с зарплатой 115269
Ведущий программист с опытом Python с зарплатой 173388
Инженер - программист АСУ ТП с опытом Python с зарплатой 194213
Инженер-программист (Клинский филиал) с опытом Python с зарплатой 105595
Инженер-программист (Орехово-Зуевский филиал) с опытом Python с зарплатой 182762
Инженер-программист 1 категории с опытом Python с зарплатой 126705
Инженер-программист ККТ с опытом Python с зарплатой 142527
Инженер-программист ПЛИС с опытом Python с зарплатой 173492
Инженер-программист САПОУ (java) с опытом Python с зарплатой 191113
Инженер-электронщик (программист АСУ ТП) с опытом Python с зарплатой 108254
Помощник веб-программиста с опытом Python с зарплатой 102927
Системный программист (C, Linux) с опытом Python с зарплатой 151216
Старший программист с опытом Python с зарплатой 125762
инженер - программист с опытом Python с зарплатой 123511
инженер-программист с опытом Python с зарплатой 166679
педагог программист с опытом Python с зарплатой 178105

Block has been executed during 0.0665019 s
```